

# COLD FUSION Developer's Journal

ColdFusionJournal.com

February, 2000 Volume: 2 Issue: 2



Editorial

## Most Wonderful Time of the Year

Robert Diamond page 5

## SQL Corner Generic SQL Table Export

David Schwartz page 22

## Book Review Sams Teach Yourself SQL in 10 Minutes


Emily Kim page 44

## Tips & Techniques ColdFusion & Generator Stock Charts

Andrew Stopford page 46

CFDJ NEWS  
page 54

**SYS.CON**  
PUBLICATIONS



Develop a set of custom tags that allow you to turn jumbled data pages into user-friendly wizards

## CFDJ Feature: Improving Web Page Usability with Wizards

*Using wizards can really increase the usability of your applications*



6

Daniel Jean

## CFDJ Feature: The Truth About ColdFusion and DCOM

*Using them together opens up new possibilities for your Web applications*



14

Elias K. Jo &  
Daniel Del Savio

## <BF> on <CF>: Preserve Precious Resources - Recycle

*It's good for the development environment*



18

Ben Forta

## CFDJ Special Feature: ColdFusion Basics

*A book excerpt on dynamic page development—from the guru*

30

Ben Forta

## Journeyman ColdFusion: Hidden Gems in 4.0.1

*What you might have missed*



40

Charles Arehart

## Foundations: A Fusebox How-To

*Using Fusebox to gather requirements*



48

Hal Helms

# AbleCommerce



The flexible solution for building electronic shopping sites

## AbleCommerce



### CLIENT LIST

A few of our  
shopping customers

24 Hour Fitness  
Bushnell  
Johns Hopkins University  
BOC Gases  
Key Tronic  
Casio  
Citgo  
GSA  
MADO  
Lucky Brand Jeans  
Compaq  
Mercury Marine  
Dairy Farmers of America  
Mitsubishi  
FAA  
Nike  
Garth Brooks  
Miss Universe/USA/Teen USA  
NorWest Financial  
George Strait  
PGE  
Edwin Watts Golf Shops  
Qwest  
Hasbro  
Siemens  
U.S. Department of Treasury  
International Paper  
United Nations  
Jimmy Buffett  
Jeff Gordon  
Verlo  
Swiss Army Knives  
New England Patriots  
To see more, visit  
[www.topsites.com](http://www.topsites.com)

## Complete Site Building and Administration from your browser.

Build "your\_store.com" with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing. For accurate shipping use AbleShipper™ UPS auto-calculator with the flexible AbleCommerce™ shipping tables. Fulfill document and software purchases through electronic delivery. Check orders online, generate automated email and fax notification.



## Enterprise Scalability for stores of any size.

Supports Microsoft Access and SQL. Add store and server license packs to create an unlimited number of stores. Two security levels for system and merchant administration allow ISP's to empower renters with online store design and maintenance abilities.

You can also integrate and customize AbleCommerce™ with source code.



## Easy Design Interface for eye catching sites.

Upload graphics directly to your server. Use prebuilt store templates for quick design. Customize your web pages and styles dynamically. Add your own HTML, Java, Flash or any other script to create a unique look. Add comment forms and surveys with automated responses to collect data from your visitors. Track visitor click throughs to determine the strengths of your site design.



*AbleCommerce is a perfect example of ColdFusion's strength and flexibility.*

-Ben Forta, Allaire Corp.

*The system is easy to use and very flexible from both a customer and administrator standpoint.*

-Rob Derby, Casio, Inc.

TRY IT OUT FREE!

Download the 30-Day Evaluation

[www.ablecommerce.com](http://www.ablecommerce.com)

AbleCommerce, 11700 NE 95th Street, Suite 100, Vancouver, Washington, 98682, (360) 253-4142

AbleCommerce is a trademark of Able Solutions Corporation. ColdFusion is a trademark of Allaire Corp. Company or product names referenced may be the trademark or registered trademark of their respective companies. © Able Solutions Corporation, All Rights Reserved.

# Digital Nation

**[www.dedicatedserver.com](http://www.dedicatedserver.com)**

# **Datareturn**

**[www.datareturn.com](http://www.datareturn.com)**



STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,  
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

**EDITOR-IN-CHIEF** ROBERT DIAMOND  
**ART DIRECTOR** JIM MORGAN  
**EXECUTIVE EDITOR** M'LOU PINKHAM  
**PRODUCTION EDITOR** CHERYL VAN SISE  
**ASSOCIATE EDITOR** NANCY VALENTINE  
**PRODUCT REVIEW EDITOR** TOM TAULLI  
**TIPS & TECHNIQUES EDITOR** MATT NEWBERRY

### WRITERS IN THIS ISSUE

DANIEL DEL SAVIO, ROBERT DIAMOND, BEN FORTA,  
HAL HELMS, DANIEL JEAN, ELIAS K. JO, EMILY KIM,  
DAVID SCHWARTZ, ANDREW STOPFORD

### SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,  
PLEASE SEND YOUR LETTERS TO  
SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE 800 513-7111

COVER PRICE \$8.99/ISSUE

DOMESTIC \$79/YR. (12 ISSUES)

CANADA/MEXICO \$99/YR.

OVERSEAS \$129/YR

BACK ISSUES \$12 EACH

**PUBLISHER, PRESIDENT AND CEO** FUAT A. KIRCAALI  
**VICE PRESIDENT, PRODUCTION** JIM MORGAN  
**VICE PRESIDENT, MARKETING** CARMEN GONZALEZ  
**ACCOUNTING MANAGER** RELI HOROWITZ  
**ADVERTISING ACCOUNT MANAGER** ROBYN FORMA  
**ADVERTISING ASSISTANT** MEGAN RING  
**ADVERTISING ASSISTANT** CHRISTINE RUSSELL  
**GRAPHIC DESIGNER** ALEX BOTERO  
**GRAPHIC DESIGN INTERN** AARATHI VENKATARAMAN  
**WEBMASTER** BRUNO Y. DECAUDIN  
**WEB EDITOR** BARD DEMA  
**WEB SERVICES INTERN** DIGANT B. DAVE  
**CUSTOMER SERVICE** ANN MARIE MILILLO  
**JDJ STORE** COM JACLYN REDMOND  
**ONLINE CUSTOMER SERVICE** AMANDA MOSKOWITZ

### EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC. 39 E. CENTRAL AVE.,  
PEARL RIVER, NY 10965  
TELEPHONE: 914 735-7300 FAX: 914 735-6547

COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)  
IS PUBLISHED MONTHLY (12 TIMES A YEAR)  
FOR \$49.99 BY SYS-CON PUBLICATIONS, INC.,  
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306.

### POSTMASTER

SEND ADDRESS CHANGES TO:  
COLDFUSION DEVELOPER'S JOURNAL  
SYS-CON PUBLICATIONS, INC.

39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306

### © COPYRIGHT

COPYRIGHT © 2000 BY SYS-CON PUBLICATIONS, INC.  
ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE  
REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS,  
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY  
INFORMATION STORAGE AND RETRIEVAL SYSTEM,  
WITHOUT WRITTEN PERMISSION.

FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR.

SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE,  
REPRODUCE AND AUTHORIZE ITS READERS TO USE  
THE ARTICLES SUBMITTED FOR PUBLICATION.

### WORLDWIDE DISTRIBUTION

BY CURTIS CIRCULATION COMPANY 739 RIVER ROAD,  
NEW MILFORD NJ 07646-3048 PHONE: 201 634-7400

### DISTRIBUTED IN USA

BY INTERNATIONAL PERIODICAL DISTRIBUTORS  
674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075  
619 481-5928

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES  
ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS  
OF THEIR RESPECTIVE COMPANIES.

# The Most Wonderful Time of the Year



BY ROBERT DIAMOND

Okay, that title might be considered about two months late by some, but please read on. For some people the most wonderful time of the year falls in the holiday season when friends, family and – best of all – presents surround them. For ColdFusion developers, though, this time of the year is proving to be much more exciting and it's just getting started. So far we've got the release of ColdFusion 4.5 – already with it's first little bug patch released, which is definitely a rite of passage for many new products these days. Some of the most anticipated products are Allaire Spectra, which has already shipped, and ColdFusion for Linux, which is in its final beta stages and will soon be out. If the beta is any indication of what's to come, all those people who are anti-NT will soon be very happy. These new product releases from Allaire and others are just one indication of what's happening in the industry.

If traffic on the CF hot spots on the Net are any indication, the use of ColdFusion is multiplying exponentially. This is great for the seasoned developer because it means more add-ons, more tags, more support on ISPs and ultimately a greater product base to draw from while developing. The more the merrier! To add to all the exciting news and announcements in the industry, we here at **CFDJ** have some of our own as well...


### Readers' Choice Awards – Voting Has Begun!

Voting for the first annual **ColdFusion Developer's Journal** Readers' Choice Awards began on February 1 and will run until September 1. I personally am very excited about the awards, because unlike most other accolades given out in the software industry, the Readers' Choice Awards represent the true opinion of readers and developers...of you, the people whose opinions really count. Categories represented in the awards include Best Book, Best Consulting Service, Best Custom Tag, Best Database Tool, Best Design Service, Best E-Business Software, Best Education and Training, Best Testing Tool, Best Web Development Tool, Best Web Hosting, Best Web Performance Tool, Best Web Site. These categories were chosen by our editors and echoed by the companies that nominated products. They cover just about everything that's available, and the final results when announced should provide a clear view of the best of the best.

The voting form is located at [www.sys-con.com/coldfusion/readers-choice2000](http://www.sys-con.com/coldfusion/readers-choice2000). As I'm sure you'll notice immediately, the entire system was developed using ColdFusion. J Live voting updates are available on the Web site, along with graphs charting the progress. The awards will be announced and presented to the winners at the Allaire conference in September.

### Starting with This Issue...

You'll notice an ongoing feature series starting with this issue: "A Beginner's Guide to ColdFusion." Due to reader demand to cover some beginners' topics along with the advanced ones we hit on every month, we'll draw material from Ben Forta's highly successful *The Coldfusion 4.0 Web Application Construction Kit*. For CF old-timers it'll be a nice trip down memory lane or a good way to brush up on some simple concepts that sometimes can get lost in the mix. For the newbies it should be a great way to start getting up to speed, bridging the gap between beginner and advanced.

Until next time... 

*Robert Diamond*

ABOUT THE  
AUTHOR  
*Robert Diamond is  
editor-in-chief of  
ColdFusion Developer's  
Journal.*

# IMPROVING WEB PAGE USABILITY WITH WIZARDS

Develop a set of custom tags that allow you to turn jumbled data pages into user-friendly wizards

BY DANIEL JEAN

**J**ust about everyone who uses a Windows-based system is familiar with the wizard metaphor.

The wizard is a series of screens, each one asking questions and gathering data. The collected data isn't submitted until the final screen, which means the user can go backward or forward through the screens and adjust the data appropriately. Since users are already familiar and comfortable with this metaphor, using wizards can truly increase the usability of your applications.

The wizard metaphor lends itself fairly well to the Web except for two problems: (1) how to save temporary information (state) in a stateless environment, and (2) controlling the navigation logic on each wizard page. Both problems can be handled quite nicely with custom tags, structures and WDDX. The final result is a set of tags that allow you to quickly and easily turn your data-gathering pages into standardized, user-friendly wizards (see Figure 1).

## Requirements

Before we rush off to start coding, we should do some basic analysis and design work. Here's a list of requirements for the tags:

1. Setting up, initializing and using the wizard should be easy.
2. Each screen needs to be created with data-collection fields only; the tag should take care of navigation and state management.
3. Every screen should have a Cancel button.
4. Every screen but the first one should have a Back button.

5. Every screen but the last one should have a Next button.
6. The last screen should have a Finish button that calls a final processing page. At that point, all collected data should be easily accessible by the developer.
7. The temporary saving of the field values should be invisible to the developer.
8. There should be no reliance on cookies or URL variables, thus no session variables.

## How About a Strategy?

We can solve the first requirement by using the parent/child tag architecture. A good example of this is the ColdFusion `<CFHTTP>` tag. Setting up and initializing the `<CFHTTP>` tag would be tedious and complex if not for its child `<CFHTTPPARAM>` tag. Why don't we create a set of parent/child tags to collect the information needed for our wizard? We'll call the parent tag `<CF_WIZARD>` and the child tags `<CF_WIZSCREEN>` and `<CF_WIZFIELD>`.



Once we have all the necessary information, we need to get it into the developer's forms. One way to accomplish this is to simply take over the whole form by creating a replacement for the <FORM> tag. If we have control over the form, we can add navigation buttons as necessary and control their actions using JavaScript. We can also sneak some hidden fields into the form to help us save state without the user's realizing it. We'll call this tag <CF\_WIZFORM>.

We need to save two separate pieces of information from page to page. First, we need to pass a list of the screens and the order in which they should be displayed. We can do this by passing a comma-separated list of URLs in a hidden field. Second, we need to keep track of all the fields and their current values. ColdFusion uses structures to track URL variables, CGI variables and Form fields, so we might as well use a structure to track our wizard fields. Developers are familiar with this format, so let's take advantage of it and make the tag as user-friendly as possible. The one problem with using a structure is that it can't be passed easily from page to page – this is where WDDX comes in. It allows us to serialize a structure into one long string of XML (which can be passed from page to page), then deserialize that string later into the original, fully featured structure (see Figure 2).

## Let's Gather Some Data

The first step is to create a child tag that collects information about the fields we want the wizard to track – specifically, the name of the field and a default value for each field. To do this, create the file wizfield.cfm and save it to the /cfusion/customtags/ directory. Now wizfield.cfm can be accessed by calling <CF\_WIZFIELD>. For the remainder of the article I'll refer to the tag name rather than the actual file name.

The <CF\_WIZFIELD> tag consists of only three lines of code:

```
<CFPARAM Name="Attributes.Name"
    Default="">
<CFPARAM Name="Attributes.Default"
    Default="">
<CFASSOCIATE BaseTag="CF_WIZARD"
    DataCollection="FieldArray">
```

The first two lines collect a name and default for each field through the tag's attributes. The third line sets a BaseTag (or parent tag) and saves the attributes collected into the array specified, which will be available to the parent tag. The array is going to have one entry for each child tag, so if there are three <CF\_WIZFIELD> tags there will be three corresponding entries

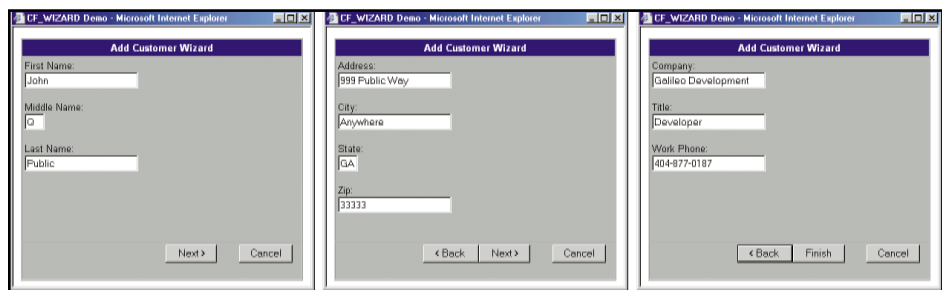


FIGURE 1: A typical wizard

in the array. Now the complex part: each entry in the array is actually a structure describing the attributes passed to that child tag. Figure 3 illustrates the array and the structures it holds.

Notice the FieldArray has three entries (since we used three <CF\_WIZFIELD> tags), and each entry has a structure describing the attributes that were passed to <CF\_WIZFIELD>.

The <CF\_WIZSCREEN> tag (create wizscreen.cfm and save in /cfusion/customtags/) is similar to the <CF\_WIZFIELD> tag except that we collect only one attribute, the URL. Also, we save the attributes in an array called ScreenArray rather than the FieldArray we specified for the <CF\_WIZFIELD> tag. This is done using the same <CFASSOCIATE> tag but we specify a different DataCollection:

```
<CFASSOCIATE BaseTag="CF_WIZARD"
    DataCollection="ScreenArray">
```

The complete code for the foregoing snippet is in Listing 1. Figure 4 illustrates the ScreenArray and its structures.

Now that both child tags are collecting data, we can create the <CF\_WIZARD> parent tag to use the data. As the parent, it has two distinct jobs. It collects the final bits of information we need and then, using its child tags, it starts the wizard process. The <CF\_WIZARD> tag gets called twice, once when it's opened and once when it's closed, so we can collect the additional information when it's opened via passed-in attributes. Then, when the tag is closed, we can start the wizard process.

The two additional pieces of information the <CF\_WIZARD> tag collects are the FinishURL and the CancelURL. These will be the URLs the wizard loads if the Finish or Cancel button is pushed. (See Listing 2 for the complete source code for the following snippet.)

```
<CFF If ThisTag.ExecutionMode IS 'start'>
    <CFPARAM Name="Attributes.FinishURL"
        Default="">
    <CFPARAM Name="Attributes.CancelURL"
        Default="#CGI.Script_Name#">
```

ThisTag.ExecutionMode is used to determine whether this is the start (or open) tag call or whether it's the end (or close) tag call. If it's the start, capture the FinishURL and CancelURL attributes, then we can set up and initialize the Fields structure and the Screens list. The two child tags collected this information for us and saved them in separate arrays. The arrays are accessed using the thisTag scope, so thisTag.FieldArray holds the <CF\_WIZFIELD> data and thisTag.ScreenArray holds the <CF\_WIZSCREEN> data. Let's continue our <CF\_WIZARD> tag by getting the FieldArray information and putting it into a Fields structure (see Listing 2 for complete source code).

```
<CFSCRIPT>
    Caller.Fields = StructNew();
    for (i=1;
        i LTE ArrayLen(thisTag.FieldArray);
        i=i+1)
    {
        FieldAttributes=thisTag.FieldArray[i];
        StructInsert(Caller.Fields,
            FieldAttributes.Name,
            FieldAttributes.Default);
    }
</CFSCRIPT>
<CFWDDX Action="CFML2WDDX"
    Input="#Caller.Fields#"
    Output="FieldWDDX">
```

Create a structure called *Fields* and use the caller scope so the calling page has access to the new structure, then set up a loop that goes through each entry in the FieldArray. Remember, this is an array of structures, so each entry holds a structure. Before getting to the actual information, we have to get the structure itself. To do this we use the FieldAttributes variable to temporarily hold each structure. Once we have the structure from this entry, we use it to insert key/value pairs into our Fields structure. The last thing we do with our field information is take our newly created structure and serialize it into a WDDX packet using the <CFWDDX> tag. Having the structure in this form will allow us to stuff it into a hidden form field and pass it along to each of our wizard pages.

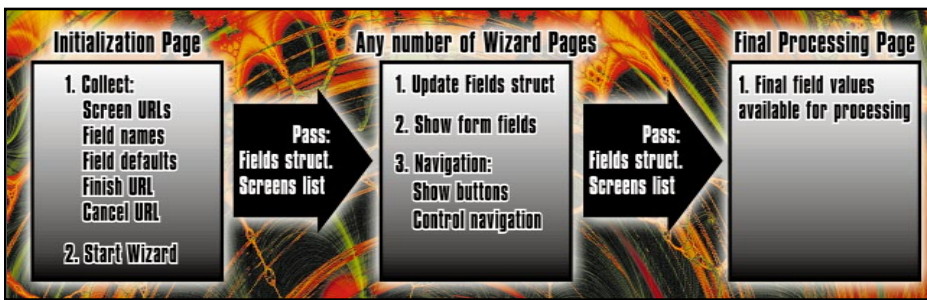


FIGURE 2 Program flow

<pre> &lt;CF_WIZARD&gt;   &lt;CF_WIZFIELD Name="FirstName" Default="John"&gt;   &lt;CF_WIZFIELD Name="MiddleInit" Default="Q"&gt;   &lt;CF_WIZFIELD Name="LastName" Default="Doe"&gt; &lt;/CF_WIZARD&gt; </pre>	
FieldArray:	
FieldArray[1]	Structure.Name="FirstName".Structure.Default="John"
FieldArray[2]	Structure.Name="MiddleInit".Structure.Default="Q"
FieldArray[3]	Structure.Name="LastName".Structure.Default="Doe"

FIGURE 3 The FieldArray

The next step in initializing the wizard is to take all the data gathered by the `<CF_WIZSCREEN>` tag and put it into an ordered, comma-separated list (see complete code in Listing 2).

```

<CFSCRIPT>
Screens = "";
for (i=1;
    i LTE ArrayLen(this sTag.ScreenArray);
    i=i+1)
{
    ScreenAttributes=this sTag.ScreenArray[i];
    Screens=ListAppend(Screens,
        ScreenAttributes.URL);
}
Screens=ListPrepend(Screens,
    Attributes.CancelURL);
Screens=ListAppend(Screens,
    Attributes.FinishURL);
</CFSCRIPT>

```

We access the ScreenArray the exact same way we access the FieldArray, except this time we create a list instead of a structure. The list is created in the order of the `<CF_WIZSCREEN>` tags, so you can change the screen order by simply changing their order. Since Screens is a list of URLs, we can prepend the CancelURL and then append the FinishURL to the list. This saves us from having to put them in their own hidden fields. Now they'll just go right along with the rest of the screen information that's passed page to page.

The last thing the `<CF_WIZARD>` tag

does is actually start the whole wizard process. This is done by creating a form, stuffing our Fields and Screens into hidden fields, and using JavaScript to "auto-submit" the form (see also Listing 2).

```

<FORM Name="WizForm" Method="Post"
    Action="#ListGetAt(Screens, 2)#">
  <INPUT Type="hidden" Name="WizFields"
    Value="#HTMLEditFormat(FieldWDDX)#">
  <INPUT Type="hidden" Name="WizScreens"
    Value="#Screens#">
  <INPUT Type="hidden" Value="2"
    Name="WizCurrentScreen">
</FORM>
<SCRIPT Language="javascript">
    document.fgColor = document.bgColor;
    document.WizForm.submit();
</SCRIPT>

```

The first thing you'll notice about the `<FORM>` is that we set the Action attribute equal to the second item in our Screens list. Remember, the CancelURL is prepended to the Screens list, so the first screen we want to show is actually the second URL in the list. Next, pack the data into hidden fields so the first screen has access to them. Notice that we use the `HTMLEditFormat` function to format the WDDX packet. Since WDDX uses XML, we have to format the packet before putting it into a hidden field; otherwise the HTML parser will be confused by the XML tags and "your results may vary." Finally, we use JavaScript to submit the form.

## Using the Data

The wizard process has been started. Now we need to control the wizard pages themselves. We've already said that, to control the navigation and state issues, we'll replace the `<FORM>` tag with our own `<CF_WIZFORM>` tag. Naturally, the `<CF_WIZFORM>` tag will be called twice, once on start and once on end, just like the `<FORM>` tag. When it's called as the starting or opening tag, we update our Fields structure to account for any changes made on the previous screen, then do our duty as a replacement for the `<FORM>` tag and create a form (see Listing 3 for complete source code).

```

<CFWDDX Action=WDDX2CFML
    Input="#FORM.WizFields#"
    Output="Caller.Fields">
<CFSCRIPT>
    for(thisKey in Caller.Fields)
    {
        if(isDefined("FORM." & thisKey))
            Caller.Fields[thisKey] =
                evaluate("FORM." & thisKey);
    }
</CFSCRIPT>
<CFWDDX Action="CFML2WDDX"
    Input="#Caller.Fields#"
    Output="FieldWDDX">

```

We deserialize our WDDX packet back into a working Fields structure, then loop through every key in Fields and determine whether there's a matching key in the Form structure. If there is, this field was on the previous page and must be updated in our Fields structure. After all the Fields have been checked and updated, we serialize Fields back into a WDDX packet so it can be passed on to the next screen. The last thing we do in the start section of `<CF_WIZFORM>` is output a `<FORM>` and create our needed hidden fields, remembering to use the `HTMLEditFormat` to save our new WDDX packet.

The main job of the end section of the `<CF_WIZFORM>` tag is to control navigation. The first step is to show only the appropriate buttons. Using the Screen list and the CurrentScreen value from our hidden fields, we determine whether or not the Back, Next or Finish buttons are displayed, then use JavaScript to determine which screen is shown next. I've put the JavaScript inline in the `<INPUT>` tag, but a function could be written that accomplishes the same thing. The JavaScript for the next button looks like this (see Listing 3 for complete code):

```

document.WizForm.action=
    '#ListGetAt(FORM.WizScreens,
        FORM.WizCurrentScreen + 1)#!';

```



# **Ektron**

**[www.ektron.com](http://www.ektron.com)**

```
document.Wi zForm.Wi zCurrentScreen.val ue =
#eval uate(' FORM.Wi zCurrentScreen + 1')#
```

If the Next button is pushed, the next screen in the Screens list should be displayed. We determine the URL by getting the CurrentScreen + 1 from the Screens list. Then, by setting the action property of the form, we force the form to load the next screen. If the Back button is pressed, the same code is used except we get the element at CurrentScreen - 1. The JavaScript for the Cancel and Finish buttons simply uses the first and last URLs in the Screens list, since that's where we stored them in the <CF\_WIZARD> tag.

## Our First Wizard

Our first requirement states that these tags have to be quick and easy to use, but so far they seem complex and difficult. In fact, we've done quite a few coding tricks and used some advanced ColdFusion technologies such as WDDX and structures. However, this is the beauty of custom tags. The user of the custom tag can get all the benefits of the tag without understanding or even knowing about the complexity built into it. To prove this, let's take our newly created custom tags and discover how easy they are to use.

### 1. Create an initialization page.

```
<HTML><BODY>
<CF_WIZARD Fini shURL="ShowUser.cfm">
  <CF_WIZFIELD Name="First Name"
    Default t="John">
  <CF_WIZFIELD Name="Last Name"
    Default t="Doe">
  <CF_WIZFIELD Name="Address"
    Default t="">
  <CF_WIZFIELD Name="Phone"
    Default t="">
  <CF_WIZSCREEN URL="GetName.cfm">
  <CF_WIZSCREEN URL="GetAddress.cfm">
</CF_WIZARD>
</BODY></HTML>
```

This page describes all the fields we want to collect and their default values. It also describes the screens and the order in which they'll be shown. Save this file as GetUserStart.cfm.

### 2. Create your data-collection pages.

```
<HTML><BODY>
<CF_WIZFORM>
<CFOUTPUT>First Name:
<INPUT Name="First Name" Type="Text"
  Val ue="#Fields.First Name#"><BR>
Last Name:
<INPUT Name="Last Name" Type="Text"
```

ScreenArray	Structure.URL
ScreenArray[1]	"/wiz/UserName.cfm"
ScreenArray[2]	"/wiz/UserAddr.cfm"
ScreenArray[3]	"/wiz/UserWork.cfm"
ScreenArray[4]	"/wiz/Other.cfm"

FIGURE 4 The ScreenArray

FIGURE 5 Our first wizard

```
Val ue="#Fields.LastName#"><BR>
</CFOUTPUT>
</CF_WIZFORM>
</BODY></HTML>
```

This page displays the form fields, but instead of using the regular <FORM> tag, use the <CF\_WIZFORM> tag. Save this file as GetName.cfm, then create another file called GetAddress.cfm. Change the <INPUT> tags to capture the Address and Phone fields (set their name to Address and Phone, then set their values to #Fields.Address# and #Fields.Phone#, respectively). Also, change the captions before the <INPUT> to reflect an appropriate title (Address and Phone, for example).

### 3. Create your final processing page.

```
<HTML><BODY>
<CF_WIZFORM>
<CFOUTPUT>
First Name: #Fields.First Name#<BR>
Last Name: #Fields.LastName#<BR>
Address: #Fields.Address#<BR>
Phone: #Fields.Phone#<BR>
</CFOUTPUT>
</BODY></HTML>
```

All the collected fields, with their final values, will be available via the Fields structure on the final processing page. In this case the final values will be displayed on the screen. Save this as ShowUser.cfm.

Now you need to call the initialization page, GetUserStart.cfm, from your browser – it's simple to set up and use (see Figure 5).

## Benefits

You don't have to look very far on the Web to find pages that are just eye candy – developers trying to use the latest technology to do the latest visual effect. However, developing real business applications means we need to look away from the eye candy and toward solutions that are user-friendly and easy to use. We can do this by using intuitive interface components (a shopping cart means buy something, a trash can means delete something) or de facto standards (red means stop, green means go). Using a de facto metaphor such as the wizard makes users feel comfortable because they instantly understand how to use the interface and what to expect from the application. Comfort level increases, usability increases, understanding increases – and if you use these tags your coding time decreases. Sounds like a winner to me!

### ABOUT THE AUTHOR

Daniel Jean is the CTO and cofounder of Galileo Development Systems, a provider of Web-based software solutions for the consulting and contract staffing industries. Daniel has worked in the client/server and Web-development industry for 10 years.

djean@galileodev.com

# **Interact**

**[www.interact.com](http://www.interact.com)**



```
<CFPARAM Name="Attributes_UIP1" Default="">
```

```
<CFPARAM Name="Attri butes. URL" Defaul t="">
<CFASSOCI ATE BaseTag="CF_WI ZARD"
      DataCol l ecti on="ScreenArray">
```

```

LISTING 2: Wizard.cfm
<CFIF ThisTag.ExecutionMode is 'start'>

```

```
<CFIF ThisTag.ExecutionMode is 'start'>
<CFPARAM Name="Attributes.FinishURL" Default="">
<CFPARAM Name="Attributes.CancelURL"
Default="#CGI.Script_Name#">
<CFSET thisTag.FieldArray = arrayNew(1)>
<CFELSE><!-- This is </CF_WIZARD> --->
<!-- Setup Fields struct and convert to WDDX --->
<CFSCRIPT>
    Caller.Fields = StructNew();
    for (i=1; i LTE ArrayLen(thisTag.FieldArray);
        i=i+1)
    {
        FieldAttributes = thisTag.FieldArray[i];
        StructInsert(Caller.Fields,
            FieldAttributes.Name,
            FieldAttributes.Default);
    }
</CFSCRIPT>
<CFWDDX Action="CFML2WDDX" Input="#Caller.Fields#"
Output="#FieldWDDX">
<CFSET na = StructClear(Caller.Fields)>
<!-- setup Screens list --->
<CFSCRIPT>
    Screens = "";
    for (i=1; i LTE ArrayLen(thisTag.ScreenArray);
        i=i+1)
    {
        ScreenAttributes = thisTag.ScreenArray[i];
        Screens = ListAppend(Screens,
            ScreenAttributes.URL);
    }
    Screens = ListPrepend(Screens,
        Attributes.CancelURL);
    Screens = ListAppend(Screens,
        Attributes.FinishURL);
</CFSCRIPT>
<!-- Autosubmitting form with hidden fields --->
<CFOUTPUT>
<FORM Name="WizForm" Method="Post"
Action="#ListGetAt(Screens, 2)#">
<INPUT Type="hidden" Name="WizFields"
Value="#HTMLEditFormat(FieldWDDX)#">
<INPUT Type="hidden" Name="WizScreens"
Value="#Screens#">
<INPUT Type="hidden" Name="WizCurrentScreen"
Value="2">
</FORM>
<SCRIPT Language="javascript">
    document.fgColor = document.bgColor;
    document.WizForm.submit();
</SCRIPT>
</CFOUTPUT>
</CFIF>
```

LISTING 3: WizForm.cfm

```
<CFSET ButtonWidth = 90>
<CFIF ThisTag.ExecutionMode is 'start'>
  <CFSET Caller.Fields = StructNew()>
  <!-- deserialize wddx into Field struct -->
  <CFWDDX Action=WDDX2CFML Input="#FORM.WiZFiElDs#"
    Output="Caller.Fields">
  <!-- Update Fields struct for all fields
    submitted via FORM -->
  <CFSCRIPT>
  for (thisKey in Caller.Fields)
```

```
{
    if(isDefined("FORM." & thi sKey))
        Cal l er. Fi el ds[thi sKey] = eval uate("FORM." &
                                                    thi sKey);
}
</CFSCRIPT>
<!-- serialize Fields structure back into wddx
      packet and then hide in hidden field -->
<CFWDDX Action="CFML2WDDX" Input="#Cal l er. Fi el ds#"
      Output="Fi el dWDDX">
<CFOUTPUT>
<FORM Name="Wi zForm" Method="Post">
<INPUT Type="hi dden" Name="Wi zFi el ds"
      Val ue="#HTMLEdi tFormat(Fi el dWDDX)#">
<INPUT Type="hi dden" Name="Wi zScreens"
      Val ue="#FORM. Wi zScreens#">
<INPUT Type="hi dden" Name="Wi zCurrentScreen"
      Val ue="0">
</CFOUTPUT>
<CFELSE><!-- Thi s i s </CF_WI ZFORM> --->
<!-- Add appropriate navigation buttons --->
<CFOUTPUT>
<TABLE border=0 cel l spac ing=0 cel l padd ing=0>
<TR><TD wi dth=#ButtonWi dth# val ign=mi ddl e
      al ign=ri ght&nbsp;   </CFOUTPUT>
<CIFI F FORM. Wi zCurrentScreen GT 2>
      <CFOUTPUT><INPUT Type="Submi t" Name="Wi zBack"
          Val ue=" < Back " AccessKey="B"
onCl ick="document. Wi zForm. act ion=' #Li stGetAt (FORM. Wi zScreens,
FORM. Wi zCurrentScreen - 1) #' ;
document. Wi zForm. Wi zCurrentScreen. val ue =
#eval uate(' FORM. Wi zCurrentScreen - 1 ')#"></CFOUTPUT>
</CIFI F>
<CFOUTPUT></TD><TD wi dth=#ButtonWi dth#
      val ign=bottom
      al ign=le ft></CFOUTPUT>
<CIFI F FORM. Wi zCurrentScreen IS
      (Li stLen (FORM. Wi zScreens ) - 1)>
      <CFOUTPUT><INPUT Type="Submi t" Name="Wi zFi ni sh"
          Val ue=" Fi ni sh " AccessKey="F"
onCl ick="document. Wi zForm. act ion=' #Li stLast (FORM. Wi z-
Screens) #' ;
document. Wi zForm. Wi zCurrentScreen. val ue =
#eval uate(' FORM. Wi zCurrentScreen + 1 ')#"></CFOUTPUT>
<CFELSE>
      <CFOUTPUT><INPUT Type="Submi t" Name="Wi zNext"
          Val ue=" Next > " AccessKey="N"
onCl ick="document. Wi zForm. act ion=' #Li stGetAt (FORM. Wi zScreens,
FORM. Wi zCurrentScreen + 1) #' ;
document. Wi zForm. Wi zCurrentScreen. val ue =
#eval uate(' FORM. Wi zCurrentScreen + 1 ')#"></CFOUTPUT>
</CIFI F>
<CFOUTPUT>&nbsp;   </TD><TD>
<TD wi dth=#ButtonWi dth# val ign=mi ddl e
      al ign=center>
<INPUT Type="Submi t" Name="Wi zCancel "
      Val ue=" Cancel " AccessKey="C"
onCl ick="document. Wi zForm. act ion=' #Li stFi rst (FORM. Wi z-
Screens) #' ;
document. Wi zForm. Wi zCurrentScreen. val ue = 2">
      </TD></TR></TABLE></FORM>
</CFOUTPUT>
<CFSET na = StructCl ear(Cal l er. Fi el ds)>
</CIFI E>
```

DDDDDDDDDDDDDD

The code listing for  
this article can also be located at  
[www.ColdFusionJournal.com](http://www.ColdFusionJournal.com)

# **Shift4**

**[www.shift4.com](http://www.shift4.com)**

# The Truth About ColdFusion and DCOM

Using them together opens new windows of possibilities for your Web applications

BY  
ELIAS K.  
JO  
&  
DANIEL  
DEL SAVIO



We'll view DCOM through ColdFusion and cover some of the features and pitfalls of using these technologies together. The examples here are made in conjunction with Visual Basic's support of the distributed interface. By the end of this article you should be able to start planning the framework for your DCOM-enabled ColdFusion applications.

DCOM is a derivative of COM, Microsoft's interface definition standard. COM provides a medium for developers to create interoperable interfaces and a means to connect to any COM-compliant service, regardless of implementation language or process space. DCOM extends COM functionality by providing distributed services such as transport, security and directory services that allow component interfaces to be accessed anywhere across a network.

## How ColdFusion Accesses DCOM Services

ColdFusion provides a mechanism to access COM components through its CFOBJECT tag, which is used to gain the initial reference for an external object through COM, DCOM or CORBA. (COM and DCOM are used synonymously in reference to the CFOBJECT tag because in ColdFusion these types of objects are identical.)

```
<CFOBJECT type="COM" class=""
action="create" name="xxx">
```

Keep in mind that COM components are designated with a unique ID, known as their GUID, each time they're compiled. Components are registered when the GUID is incor-

Extending ColdFusion to incorporate Microsoft's Distributed Component Object Model (DCOM) is a powerful way to expand the capabilities of the application server on the Microsoft platform.

porated into the Windows registry. The same GUID version must be registered on the calling and receiving machine for the DCOM services to be properly located.

## Building DCOM Components with Visual Basic

DCOM components can be built using various programming languages (Java, Visual Basic, Visual C++, etc.). In this article we'll cover the specifics of how to build and deploy COM objects written in Visual Basic. When compiling a Visual Basic 6.0 COM object for compatibility with the distributed model architecture, set an additional compilation option in your project's properties – the Remote Server Files option on the Component Tab of the Project Properties sheet.

When the time comes to deploy your compiled DCOM object, it's recommended you use Microsoft's Package & Deployment Wizard to create an installation program for the object. This wizard provides a simple means for placing and registering your new DCOM component. Remember to install the new component on any client and server machines that will be using this object. The last step in setting up your DCOM component is to configure its registered settings through the utility DCOMCNFG.exe provided with Microsoft's Visual Studio 6.

In using Microsoft's DCOM configuration utility (DCOMCNFG.EXE), configuring your newly registered DCOM objects is as simple as a few clicks of the mouse. Launch the "dcomcnfg" utility through your Windows command prompt and find your registered object in the list

of registered components. Select the new component and click the Properties button for it. The utility must be used to configure the client and server machines where the component is installed. The following are recommendations for how to initially set your DCOM configurations on each machine.

On the client machine:

- **Location tab:** Choose *only* "Run Application on following server." Specify server in textbox.
- **Security tab:** Set permissions for who can launch and access your component.
- **Identity tab:** Choose "The launching user."
- **End points:** Leave defaults.

On the server machine:

- **Location tab:** Choose "Application on this computer."
- **Security tab:** Give all necessary permissions. DCOM won't work with incorrect permissions.
- **Identity tab:** Choose "The interactive user."
- **End points:** Leave defaults.

## Passing Variables from ColdFusion to DCOM

This process has some intricacies that we'll cover in respect to Visual Basic components. To support the interoperability of accessing COM components, we use a process called *marshaling* to pass variables to COM. It's the process of deconstructing objects and data types from one implementation and translating them into another.

In DCOM components, functions accept two types of variables: primitive and object references. Primitive data types refer to vari-



# **Eprise**

**[www.eprise.com](http://www.eprise.com)**

## ABOUT THE AUTHORS

Elias K. Jo is a consultant with Inventa Corporation, a professional services firm focused on B2B e-commerce integration. He has five years of programming experience with client/server and multitier application projects and is a Sun-certified Java Programmer.

*Daniel Del Savio, a graduate of the University of Notre Dame, is also a consultant with Inventa Corporation. He's been involved with the development of several multitiered and data processing applications.*

ables that are string, integers, booleans and real number. When passing primitive data types to a COM interface, that interface must be defined to accept the value with the “ByVal” attribute. When passing nonprimitive variables or object variables through COM, the reference of the variable is passed and no additional attribute needs to be set. For example, the following code snippet shows how to declare a Visual Basic function that’s able to accept the Integer primitive data type.

```
Function TestFunction(By Val Parameter1 As Integer)
```

End Function

## Managing DCOM Objects in Multiuser Scenarios

Two main issues must be considered when programming your DCOM-enabled ColdFusion application for multiuser scenarios. The first is that the instantiation of remote objects tends to be resource-intensive and can have performance implications for your application. The second is the complication of handling multiple concurrent requests to your remote object. Because ColdFusion handles user requests in multiple

threads and the ColdFusion server itself can be loaded on multiple machines (via load balancing), the developer must consider whether the DCOM objects are thread-safe. Using resource pools to manage your remote components can help you through these problems. Resource pools are a programming paradigm generally used to manage a limited number of resources for multiuser services.

In ColdFusion you can implement a resource pool manager by following this example: enable application-level variables through the `Application.cfm` file, then add the following two application variables.

Appl i cati on. cfm:

```
<cfparam name="appli cati on. recPool "
default t="NewArray(1)">
<cfparam
name="appli cati on. MAX_POOL_SI ZE"
default t=5>
```

These variables are used to hold the resources and define how many are to be pooled. The maximum number of resources should be defined in terms of how resource-intensive the components are and how robust the server is that will be running them.

Listing 1 is the algorithm for actually implementing an example

pooling mechanism. As you'll notice, this algorithm assumes that the DCOM component is defined with the "isAvailable" and "setAvailability" functions. Use this snippet to create a custom tag to manage your resource pools.

The algorithm will loop through the pool until a resource becomes available. It's important that requesters for pool resources set them to available once they're done. The resource pool paradigm is good for handling resources that are used quickly and then returned. It doesn't work well when they're used for an extended period of time, in which case circumstances of deadlock can occur. The resource pool also solves any concurrency issues because it ensures that no more than one user request accesses any given resource at the same time.

Using ColdFusion and DCOM together opens new windows of possibility for your Web applications. However, the many intricacies involved can create hours of trial and error. With this framework of knowledge, we hope that using DCOM can be a viable and painless solution for expanding your Web application.

ELIAS.JO@INVENTA.COM

DDELSAVIO@INVENTA.COM

### LISTING 1

```

<!-- myResource is the variable that is to be set by
the requesting user
-->
<cfparam name="pool Counter" default="1">

<!-- Loop until the variable myResource is set -->
<cfloop condition="Not IsDefined("myResource")">
  <!-- If a resource variable is defined then check
  its availability -->
  <cfif pool Counter Lt ArrayLen(application.recPool)>

    <cftry>
      <!-- if the object is available, set myResource.
      If not just
          continue to loop. -->
    <cfset tempResource = application.recPool[pool Counter]>
    <cfif tempResource.isAvailable() is "TRUE">
      <cfset tempResource.setAvailability("FALSE")>
      <cfset myResource = tempResource>

      <!-- The user request is now responsible
      to setAvailability
      back to TRUE. -->

    </cfif>
  <cfcatch type="ANY">
    <!-- If the object method fails, the remote
    object reference
    may have been lost, try to recreate a
    new object -->
  </cfcatch>
</cfloop>

```

```

<CFOBJECT type="COM" class="Elias.testDCOM"
    action="create"
name="myResource">
    <cfset application.recPool [pool Counter] =
        myResource>
    </cfcatch>
</cftry>
<cfelse>
    <!--- Create an object if one is not defined for
        a pool location.
        Because it is a new object, the check
            for availability is
            Not necessary. --->
    <CFOBJECT type="COM" class="Elias.testDCOM"
        action="create"
name="myResource">
    <cfset application.recPool [pool Counter] = myResource>
    <cfset myResource.setAvailability("FALSE")>
</cffi>
<!--- Increment the pool counter to the next resource
    in the pool --->
<cfset poolCounter = poolCounter + 1>
<cffi poolCounter gt application.MAX_POOL_SIZE>
    <cfset poolCounter = 1>
</cffi>
</cfoop>

```

COD  
LISTING

CODE  
LISTING

The code listing for  
this article can also be located at  
[www.ColdFusionJournal.com](http://www.ColdFusionJournal.com)

# **SD 2000**

**[www.sdexpo.com](http://www.sdexpo.com)**



# Preserve Precious Resources—Recycle

BY  
BEN  
FORTA



Recently I gave my personal Web site ([www.forta.com](http://www.forta.com)) a much-needed and long overdue overhaul.

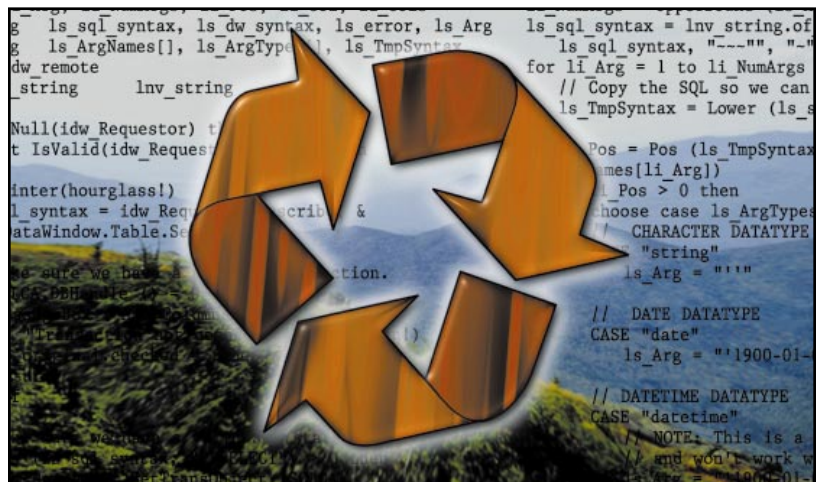
Doing so gave me the chance to clean up lots of old code (updating it to use new CF features in the process) while rethinking the organization and management of code to facilitate better reuse of common code and components.

Code organization and reuse are important topics, and judging by lots of code I've seen at customer sites of late (production code nonetheless), they are topics that need clarifying and addressing. So I'd like to use this month's column to share some of these ideas with you.

## Why Reuse?

Programmers love reusing code, and for good reasons.

- Reusing code can dramatically cut down development time. After all, why develop something from scratch when you can reuse something written previously?
- Reusing proven code helps write bugfree applications. Programmers are human. We make mistakes, and we gradually fix them when we find them. Why start with a whole new set of mistakes when you can leverage the code with mistakes already fixed? Furthermore, applications that share common code also share common code fixes.
- Code that's properly shared and reused is far more maintainable. If you've ever had to update menu options (images, links, DHTML code) in multiple application pages you know what I mean.
- Code designed for reuse can be shared far easier. If another developer needs to use your complex drop-down menu code, invoking an abstracted menu



object is far simpler than nitpicking and dissecting an application page with that code embedded in it.

- And finally, the very act of writing code with reuse in mind helps organize code properly.

Well-organized code, code broken into small, bite-size chunks, code thought through and designed to be reusable – that's the kind of code that separates experienced professional developers from beginners.

That's true of any language, and CFML is no different. In fact, CFML has several language features designed with code organization and reuse in mind.

## The <CFINCLUDE> Tag

ColdFusion's <CFINCLUDE> tag has been part of CFML for a long time now. It is used to perform server-side includes. When ColdFusion encounters a <CFINCLUDE> in your code, it includes the appropriate CFM file right then and there at the location of the <CFINCLUDE>, as if the included

code had been typed there directly.

And that's important to understand. ColdFusion processes the included file as if it were part of the calling page. This means that the two pages share the same scope, so any variables or queries defined before the <CFINCLUDE> are visible (can be read and written to) within the included code. Similarly, any variables or queries defined within the included page are visible in the calling page to any code after the <CFINCLUDE> tag.

This also means that there is no protection of data. Within an included page you can (deliberately or inadvertently) change or overwrite data in the caller page. In fact, as <CFINCLUDE> has no formal mechanism for passing data (parameters or attributes) to included code, setting variables explicitly with <CFSET> before a <CFINCLUDE> is often necessary.

This is a very important behavior to bear in mind when using <CFINCLUDE>, and it is not an oversight or bug. Rather, this behavior is intended and by design. Because included pages share the same

# Allaire

**[www.allaire.com](http://www.allaire.com)**

scope, ColdFusion's overhead in including files is minimal. This translates into extremely fast performance. In fact, you'll notice no real difference in execution time between a single thousand-line page and 10 included pages of 100 lines each.

So when should (and when shouldn't) <CFINCLUDE> be used? Obviously, if all you're doing is grabbing content to be sent to the client, a <CFINCLUDE> makes a lot of sense. For example, my site uses a collection of JavaScript functions that I want available to most pages. Rather than copy all that JavaScript into each file, I simply insert the following line into each file:

```
<CFINCLUDE
TEMPLATE="/common/jsfuncs.cfm">
```

This way, ColdFusion inserts the entire jsfuncs.cfm file into all the pages that need it.

But if your included code does any form of CFML-based programmatic processing or manipulation, it might not be a candidate for use with <CFINCLUDE>. For example, I have a block of code that surrounds the body of each page, code that contains header and title information, menus and toolbars, and DHTML-related code. If this code had been static, <CFINCLUDE> use would be appropriate. But my code isn't static at all; it has all sorts of conditional code in it, like logic to highlight the currently selected menu option. Code like that, code that manipulates data (and can thus inadvertently manipulate data it shouldn't), code that needs data passed to it, code that does more than just include content to be sent to the client – that kind of code should not be included with <CFINCLUDE> at all.

To summarize, <CFINCLUDE> is fast, it's easy to use and it's ideally suited for use with simple code and content.

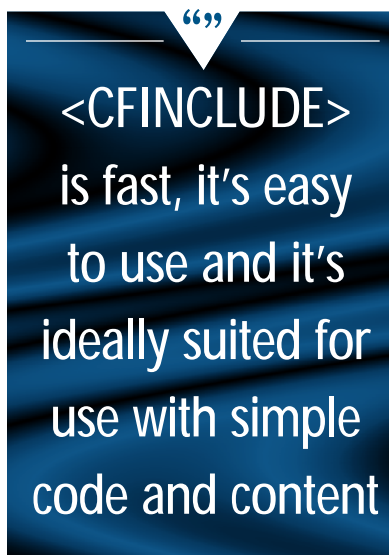
### Custom Tags

Custom Tags were introduced in ColdFusion 3, and substantially enhanced in ColdFusion 4. These tags provide a mechanism to execute code without actually including it in your page. A Custom Tag runs in its own scope, so any

queries or variables defined within it aren't automatically visible to the calling code. Similarly, data defined in the calling code is not automatically visible to code within the Custom tag.

Unlike <CFINCLUDE>, Custom Tags have a formal method for passing data to called code. Like any other CFML tag, these tags can take attributes, optional parameters passed in the standard NAME="VALUE" format. So while data can indeed be passed to Custom Tags, this isn't an automatic process. You must pass attributes explicitly if they are to be visible with the Custom Tag.

Notice that I used the word *automatic*. It is indeed possible to expose calling page data to the Custom Tag by using the CALLER scope. There is



even a special scope (introduced in ColdFusion 4.01) called REQUEST that is automatically shared between calling pages and Custom Tags. But CALLER and REQUEST aren't the default scopes, and using them is your choice. In other words, Custom Tags provide data protection unless you explicitly choose to violate it. (The exception is read-only data, like CGI variables, URL parameters or form fields, which are automatically visible to code within Custom Tags because there's no risk in overwriting them anyway.)

And there's more. Custom Tags can be paired (a start and an end tag), and they can also be nested (parent and child tags). This makes it possible to write extremely powerful (and complex) reusable components.

All this extra power is a good thing, but it's also expensive. ColdFusion has to do a lot more work to execute Custom Tags while providing the protection of separate scopes, and that translates into slower execution time. Custom Tags don't execute as quickly as <CFINCLUDE>. That's just the way it is, and it's a reality to consider when using them. That doesn't mean you shouldn't use Custom Tags. The slight performance degradation is a small price to pay for all that flexibility and power, but at the same time, if something can be included with <CFINCLUDE>, calling it as a Custom Tag is a waste of resources.

When should (and shouldn't) Custom Tags be used? Well, if all you're doing is including a file of JavaScript functions (like the example above), or embedding a common header or footer, or including a set of CSS styles, you should be using <CFINCLUDE>. But if you're writing reusable components, files that contain lots of programmatic and conditional code, components that need to create and manipulate data without the fear of overwriting calling data, code that needs to process runtime-specified attributes, then you should definitely be writing Custom Tags.

The code I referred to above – the code that creates the page layout, including styles and menus with correctly highlighted sections – that code is written as a Custom Tag. Here's a snippet showing how that code is called:

```
<!-- Page settings -->
<CFSCRIPT>
page=StructNew();
page.title='Forta.com - About Me';
page.section='about';
...
</CFSCRIPT>

<!-- Page template -->
<CFMODULE
TEMPLATE=".. /common/SitePage.cfm"
PAGE_TITLE="#page.title#"
SECTION_ID="#page.section#"
ti on#">
...
</CFMODULE>
```

First I use a <CFSCRIPT> block to define the variables needed in this page. I could have used <CFSET>

To summarize, Custom Tags are powerful and flexible, and are ideally suited for use with more complex reusable code and components, especially components that are runtime configurable via passed attributes.

There's one additional and related topic to discuss – the reuse of data (as opposed to code). I am a firm believer in not hard-coding anything in your code. Even data sources, for example, should be passed to <CFQUERY> as a previously defined variable rather than as a string containing the actual datasource name. This way, if a data source needs to change (perhaps when migrating the application from development to staging server), a single variable needs to change, not every <CFQUERY> tag.

add code similar to the following  
to your APPLICATION.CFM:

Then you could refer to DATA-SOURCE="#dsn#" in every <CFQUERY>, and just change that single variable to change all <CFQUERY> tags.

Listing 1 is a snippet out of my new APPLICATION.CFM.

Again, I use a `<CFSCRIPT>` block for simplicity's sake and define a single structure for all my global variables. This reduces the risk of creating naming conflicts or mistakenly overwriting variables. The

This global application structure isn't used just for data sources and menus. For example, I load color and stylesheet information as well as lists of states and provinces into it. Anything that doesn't change (or changes infrequently enough that the structure can be updated as needed) can, and should, go here. (I know of some very high visibility e-commerce sites that load entire product catalogs into structures like these, thereby eliminating lots of database access.)

Writing applications with code (and data) reuse takes a little extra planning upfront. But the effort is worth it. You'll end up with better quality, and less buggy and far more manageable code as well as far more scalable code – code you'll be much happier with and proud of.





# Generic SQL Table Export

Add this export to your next application



BY  
DAVID  
SCHWARTZ

In my last article (*CFDJ*, Vol. 1, issue 6) I presented *Array\_Table*, a universal browser-based table viewer and query engine. *Array\_Table* allows you to browse, edit and query any Microsoft SQL Server table. It's 100% dynamic so it works with any database.

Many readers and clients loved the new capability and ease it offered. And they wanted more, of course.

## Feeling Trapped?

Most data-driven Web sites are located on remote servers. If you use a server-based database like Microsoft SQL Server, it's difficult to retrieve table data locally. Downloading entire tables for use with your desktop software or to keep a local backup seems impossible. In addition, your end users may periodically need to download data from the data-driven sites you create.

## Freedom At Last!

We're going to add generic table export capability to *Array\_Table*.

It'll allow you to generate a common, popular and generic text file that can be used in almost any desktop application or database. *Array\_Export* will create a text file and optionally e-mail it to you. With this functionality you'll have the freedom to actually get to your data. You can see *Array\_Table* and *Array\_Export* in action at [www.arrayone.com/Table/](http://www.arrayone.com/Table/).

## Do it!

To present your users with a list of available tables, I created *Table\_List.cfm* (see Figure 1). This form shows you a list of all the tables in a data source with the option to view the data or query the table. I've added a choice, "Export", to the table list. This is the only necessary addition to *Table\_List.cfm* (see Listing 1) and it's simply a hot link to the table export program. Here's the code....

```
<a
href="Export/Export_Menu.cfm?Table-
Name=#Table_List.Name#">Export</a>
```

All of the export files are being stored and called from a subfolder, "Export". This keeps your files neat and provides one point of reference if you have to modify any export-related code.

When you click on "Export", the *Export\_menu.cfm* script is called and you're presented with the export menu (see Figure 2 and Listing 2). The table you selected is automatically passed to the export menu. Now you have the option of exporting the data to a comma- or tab-delimited file, with or without field names in the file header.

If you enter your e-mail address, *Array\_Export* will e-mail the exported data file to you.

## How'd He Do That?

It's easier than you think. *Array\_Export.cfm* calls *Export.Cfm* (see Listing 3). *Export.Cfm* passes the tablename along with your export selections from the menu to the main ColdFusion script, *Table\_Export.cfm* (see Listing 4). This script handles all the export functionality. Let's examine *Export.Cfm*.

```
<!-- Export.CFM - Passes logon and
user choice parameters to the export
engine, Table_Export.CFM. -->
<!-- Set export file name -->
<cfset Dir_Name = #getdirectory
frompath(gettemplatepath())#>
<cfset FileName =
"#Dir_Name#\Data\#TableName##DateFor-
mat("#Now()#", "mmddyyyy")#.txt">
```

```
<!-- Call export engine -->
<cf_Table_Export
DSN="Demo"
Username="Guest"
Password="Guest"
Table="#TableName#"
FileName="#FileName#"
Delimiter="#Form.Delimiter#"
ColumnHeader="#Form.ColumnHeader#"
Email="#Form.Email#">
```

*Export.cfm* sets the directory where the exported data file will be stored. Then it sets the name of the exported file to be the file name plus the current date. So if you were exporting a table called "Sales" on December 6, 1999, the exported file name would be "Sales12061999.txt". *Export.cfm* then calls *Table\_Export.cfm* and passes all the necessary

Table Selection

View 

Table	
-------	--

 Edit? 

No	
----	--

 Structure? 

No	
----	--

	Name	Created	Owner	Rows	Export		
1	<table><tr><td></td><td>authors</td></tr></table>		authors	08/15/1997	dbo	23	<a href="#">Export</a>
	authors						
2	<table><tr><td></td><td>discounts</td></tr></table>		discounts	08/15/1997	dbo	3	<a href="#">Export</a>
	discounts						
3	<table><tr><td></td><td>employee</td></tr></table>		employee	08/15/1997	dbo	43	<a href="#">Export</a>
	employee						
4	<table><tr><td></td><td>jobs</td></tr></table>		jobs	08/15/1997	dbo	14	<a href="#">Export</a>
	jobs						
5	<table><tr><td></td><td>pub_info</td></tr></table>		pub_info	08/15/1997	dbo	0	<a href="#">Export</a>
	pub_info						
6	<table><tr><td></td><td>pub_info</td></tr></table>		pub_info	08/15/1997	dbo	8	<a href="#">Export</a>
	pub_info						
7	<table><tr><td></td><td>publishers</td></tr></table>		publishers	08/15/1997	dbo	8	<a href="#">Export</a>
	publishers						
8	<table><tr><td></td><td>roysched</td></tr></table>		roysched	08/15/1997	dbo	86	<a href="#">Export</a>
	roysched						
9	<table><tr><td></td><td>sales</td></tr></table>		sales	08/15/1997	dbo	19	<a href="#">Export</a>
	sales						
10	<table><tr><td></td><td>sales</td></tr></table>		sales	08/15/1997	dbo	21	<a href="#">Export</a>
	sales						
11	<table><tr><td></td><td>stores</td></tr></table>		stores	08/15/1997	dbo	6	<a href="#">Export</a>
	stores						
12	<table><tr><td></td><td>titleauthor</td></tr></table>		titleauthor	08/15/1997	dbo	25	<a href="#">Export</a>
	titleauthor						
13	<table><tr><td></td><td>titles</td></tr></table>		titles	08/15/1997	dbo	18	<a href="#">Export</a>
	titles						

Do it!

FIGURE 1: *Table\_List.cfm*

# Allaire

**[www.allaire.com](http://www.allaire.com)**

parameters, such as tablename and filename.

### Table\_Export.Cfm

There are four steps in exporting the table:

1. Query the table to retrieve all of the rows.
2. Retrieve all of the column names.
3. Write the data out to a text file in delimited text.
4. Mail the exported file to the user.

#### Step 1. Query the table.

```
<cfquery name="Export"
datasource="Demo" username="Guest"
password="Guest">
  Select * from #Attributes.Table#
</cfquery>
<cfif Export.RecordCount EQ 0>
  There are no records to export
<cfabort>
</cfif>
```

Simply query the table for all records. If there are none, display a message to the user and abort. If there are, we go to the second step.

#### Step 2. Get a list of all fields.

```
<cfquery name="Field_List" data-
source="Demo" username="Guest" pass-
word="Guest">
  SELECT syscolumns.Name
  FROM sysobjects , syscolumns
  WHERE syscolumns.id = sysobjects.ID
  AND upper(sysobjects.Name) =
  '#Attributes.Table#'
</cfquery>
<!-- Put field list into an array --
-->
<cfset x = "">
<cfloop query="Field_List">
  <cfset x = x & Field_List.Name>
  <cfif Field_List.CurrentRow NEQ
  Field_List.RecordCount>
    <cfset x = x & Attributes.Delimi-
    ter>
  </cfif>
</cfloop>
```

Here we query the SQL Server system tables (see December 1999 article) to retrieve the list of fieldnames in the table. We need the field list so we can export the data and separate it by fields. The variable *x* will hold the list of fields, each field-name separated by a comma.

<Cffile> is the perfect tag for creating and editing text files. Using the action="write" option, ColdFu-

sion will create the text file if it doesn't already exist.

```
<!-- Create export file -->
<cffile action="WRITE" file="#Attri-
butes.FileName#" output="" addnew-
line="No">
```

Once the file is created, you can write the field list header to it (if the user chose to add fieldnames to the file).

```
<cffile action="APPEND"
file="#Attributes.FileName#" out-
put="#x##Chr(13)##Chr(10)#" addnew-
line="No">
```

Here, the variable *x*, which now holds the list of fields, is used.

“Now you have a scalable, dynamic tool to view, query and export data from any table”

#### Step 3. Write the data out to a text file in delimited text.

To make it easier to loop through the list of fields, I prefer to use arrays. The following command converts the fieldlist to an array called FL.

```
<cfset FL = ListToArray(x, #Attributes.
Delimiter#)>
```

A simple nested (double) loop can now be used to loop through all the records in the table and output all the data. During the loop a single line of data is created in the variable *D*. It contains all data from all fields as well as the field delimiter character. After the data line is built, it can be written to the text file using <cffile> with the action="append". Append adds new lines to the file without erasing any existing data.

```
<!-- Loop through the export query
which contains all data rows -->
```

FIGURE 2 Export menu

```
<cfloop query="Export">
  <cfset d = "">
  <!-- Now loop through the field
  list and build the row export data --
  -->
  <cfloop index="n" from="1"
  to="#ArrayLen(FL)#">
    <cfset S = SetVariable("S",
    "#FL[n]#">
    <cfset d = d &
    Trim(#Evaluate(S)#>
    <cfif n NEQ ArrayLen(FL)>
      <cfset d = d & Attributes.Delimi-
      ter>
    </cfif>
  </cfloop>
  <!-- Write record to text file --
  -->
  <cffile action="APPEND"
  file="#Attributes.FileName#"
  input="#d##Chr(13)##Chr(10)#"
  addnewline="no">
</cfloop>
```

#### Step 4. Mail the exported file to the user.

After the export loop completes the file, a new export file will be sitting on the server. If the user entered an e-mail address on the Export\_menu.cfm form, the file will automatically be e-mailed to them. It doesn't get any easier than this.

### Extension Cord, Please

Now you have a scalable, dynamic tool to view, query and export data from any table. With the new data export capability it's easy to bring data back to your desktop for backup or manipulation. Since the export generates pure text files, the data can be used in virtually any application. The possibilities are still endless.



DS@ARRAYONE.COM

### ABOUT THE AUTHOR

David Schwartz is the president of Array Software Inc., a New Jersey-based software company. Array creates global data-driven Internet and intranet Web sites using ColdFusion, Oracle, MS SQL Server and Java. David has been developing turnkey custom database software for 13 years.

# Allaire

**[www.allaire.com](http://www.allaire.com)**



# LISTING 1

<!-- Table\_List.Cfm - Sample interface to the Array\_Table tag. It allows you to choose a table and then view or query it. -->

```
<TITLE>Array Table Viewer</TITLE>
<BODY background="back.gif">
<cfform action="Call_Array_Table.cfm" method="POST" enable-
cab="Yes" enctype="application/x-www-form-urlencoded">
  <cfquery name="Table_List" datasource="Demo" dbtype="ODBC"
username="Guest" password="Guest">
SELECT distinct O.Name, O.UID, O.CRdate, U.name as Owner,
l.rows
FROM sysobjects O inner join sysusers U
on (O.uid = U.uid)
inner join sysindexes I
on (I.id = O.id)
Where O.type = 'u'
ORDER BY O.Name
</cfquery>
<cfif Table_List.RecordCount EQ 0>
  There are no tables to display
</cfif>
</cfif>
<table border="0" cell spacing="0" cell padding="0"
align="center" bgcolor="Silver">
  <tr>
    <td bgcolor="#0033CC">&nbsp;</td>
    <td colspan="7" bgcolor="#0033CC"><B><FONT SIZE="2"
COLOR="white" FACE="Arial, Helvetica">Table
Select on</FONT></B></td>
  </tr>
  <tr>
    <td bgcolor="Silver">&nbsp;</td>
    <td bgcolor="Silver">&nbsp;</td>
    <td bgcolor="Silver">&nbsp;</td>
```

```
<td bgcolor="Silver">&nbsp;</td>
<td bgcolor="Silver">&nbsp;</td>
<td bgcolor="Silver">&nbsp;</td>
<td bgcolor="Silver">&nbsp;</td>
<td bgcolor="Silver">&nbsp;</td>
</tr>
<tr>
  <td height="32" colspan="8" bgcolor="Silver"><FONT
SIZE="2" FACE="Arial, Helvetica">View
  <SELECT NAME="Action">
    <OPTION SELECTED>Table</OPTION>
    <OPTION>Query</OPTION>
    <!-- <OPTION>Export</OPTION> -->
  </SELECT>
  Edit?
  <SELECT NAME="Edit">
    <option value="No" selected>No</option>
    <option value="Yes">Yes</option>
  </SELECT>
  Structure?
  <SELECT NAME="Structure">
    <option value="No" selected>No</option>
    <option value="Yes">Yes</option>
  </SELECT></FONT>
</td>

</tr>
<tr>
  <td bgcolor="Silver">&nbsp;</td>
  <td bgcolor="Silver">&nbsp;</td>
  <td bgcolor="Silver"><B><FONT SIZE="2" FACE="Arial, Hel -
vetica">Name</FONT></B></td>
  <td bgcolor="Silver"><B><FONT SIZE="2" FACE="Arial, Hel -
vetica">Created</FONT></B></td>
  <td bgcolor="Silver"><B><FONT SIZE="2" FACE="Arial, Hel -
vetica">Owner</FONT></B></td>
```

## Fusion.

## We've got it down cold.

- ColdFusion Hosting
- SSL Secure Server
- Co-location Services
- In-house ColdFusion Developers
- Microsoft FrontPage, ASP, and NetShow
- RealAudio, RealVideo, and RealFlash Server
- Adhost Merchant, a scalable, customizable storefront
- Redundant DS-3 connections, bandwidth scalable to OC-3



[www.adhost.com](http://www.adhost.com)   [www.adhostmerchant.com](http://www.adhostmerchant.com)   [sales@adhost.com](mailto:sales@adhost.com)   888-234-6781

```

<TD BGCOLOR="silver"><B><FONT SIZE="2" FACE="Arial, Hel -
vetica">Rows </FONT></B></TD>
<TD BGCOLOR="silver"><B><FONT SIZE="2" FACE="Arial, Hel -
vetica">Export</FONT></B></TD>
<td>&nbsp;</td>
</TR>
<cfoutput query="Table_List">
<TR>
<TD BGCOLOR="silver">&nbsp;</TD>
<td align="RIGHT" bgcolor="silver"><FONT SIZE="-2"
FACE="Arial, Helvetica">#Table_List.CurrentRow#</font></TD>
<TD BGCOLOR="silver"><FONT SIZE="-2" FACE="Arial, Hel veti -
ca">
<cfif Table_List.CurrentRow eq 1>
<FONT SIZE="-2" FACE="Arial, Helvetica"><cfinput
type="Radio" name="TableName" value="#Table_List.Name#"
checked="Yes">
<cfelse>
<FONT SIZE="-2" FACE="Arial, Helvetica"><cfinput
type="Radio" name="TableName" value="#Table_List.Name#">
</cfif>
#Table_List.Name#</B></TD>
<TD BGCOLOR="silver"><FONT SIZE="-2" FACE="Arial, Hel -
vetica">#DateFormat("#Table_List.CRDate#", "mm/dd/yyyy")#</TD>
<TD BGCOLOR="silver"><FONT SIZE="-2" FACE="Arial, Hel -
vetica">#Table_List.Owner#</TD>
<td align="RIGHT" bgcolor="silver"><FONT SIZE="-2"
FACE="Arial, Helvetica">#Table_List.Rows#&nbsp;</TD>
<td align="Left" bgcolor="silver"><FONT SIZE="-2"
FACE="Arial, Helvetica"><a
href="Export/Export_Menu.cfm?TableName=#Table_List.Name#">Exp
ort</a></TD>
<td BGCOLOR="silver">&nbsp;</td>
</TR>
</cfoutput>
<TR>

```

```

<td colspan="8" align="CENTER" bgcolor="silver">
<input type="Submit" name="Submit" value="Do it!">
</TD>
</TR>
</TABLE>
</cfFORM>

```

#### LISTING 2

```

<TITLE>Array Table Export</TITLE>
<cfFORM ACTION="Export.Cfm?TableName=#Tablename#"
METHOD="POST" ENCTYPE="application/x-www-form-urlencoded">
<CENTER>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0"
bgcolor="silver">
<TR>
<TD COLSPAN="3" BGCOLOR="#0033CC"><B><FONT COLOR="white"
FACE="Arial, Helvetica">Array Export v1.0</FONT></B></TD>
</TR>
<tr>
<td colspan="3" align="CENTER">&nbsp;</td>
</tr>
<tr>
<td colspan="3" align="CENTER"><FONT SIZE="2"
FACE="arial"><cfoutput>Exporting #TableName#</cfoutput></TD>
</tr>
<tr>
<td colspan="3" align="CENTER">&nbsp;</td>
</tr>
<tr>
<td align="RIGHT"><FONT SIZE="2" FACE="arial">Field deli m -
iter</FONT></TD>
<TD>
<SELECT NAME="Delimiter">
<OPTION value="Comma" SELECTED>Comma</OPTION>
<OPTION value="Tab">Tab</OPTION>
</SELECT>

```

# RSW Software

[www.rswsoftware.com](http://www.rswsoftware.com)

```
</TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD><FONT SI ZE="2" FACE="Ari al ">Add fi el d names</FONT></TD>
<TD>
<SELECT NAME="Col umnHeader">
<OPTI ON val ue="Yes">Yes</OPTI ON>
<OPTI ON val ue="No" SELECTED>No</OPTI ON>
</SELECT>
</TD>
<TD>&nbsp;</TD>
</TR>
<TR>
<TD al ign="right"><FONT SI ZE="2" FACE="Ari al ,
Hel veti ca">email fi le to</FONT></TD>
<TD><cfi nput type="Text" name="email" val ue=""
requi red="No" si ze="15" maxl ength="40"></TD>
<TD>&nbsp;</TD>
</TR>
<tr><TD col span="3" al ign="CENTER">&nbsp;</TD></tr>
<TR>
<TD COLSPAN="3" al ign="Center"><I NPUT TYPE="SUBMI T"
NAME="Submi t" VALUE="Export i t!"></TD>
</TR>
<tr><TD col span="3" al ign="CENTER">&nbsp;</TD></tr>
</TABLE>

</CENTER>
</cFFORM>
```

### LISTING 3

```
<!-- Export.CFM - Passes logon parameters to the export engine, Table_Export.CFM. --->

<!-- Set export file name --->
<cfset Dir_Name = #getdirectoryfrompath(gettemplatepath())#>
<cfset FileName = "#Dir_Name\Data\TableName##DateFormat("Now()", "mmddyyyy").txt">

<!-- Call export engine --->
<cf_Table_Export
    DSN="Demo"
    Username="Guest"
    Password="Guest"
    Table="#TableName#"
    FileName="#FileName#"
    Delimiter="#Form.Delimiter#"
    ColumnHeader="#Form.ColumnHeader#"
    Email="#Form.Email#">
```

#### LISTING 4

```
<!-- Table_Export.Cfm - Export Engine --->
<cfparam name="AttrIBUTES.DSN">
<cfparam name="AttrIBUTES.UserName">
<cfparam name="AttrIBUTES.Password">
<cfparam name="AttrIBUTES.Table">
<cfparam name="AttrIBUTES.FileName">
<cfparam name="AttrIBUTES.Delimiter">
<cfparam name="AttrIBUTES.ColumnHeader">
<cfparam name="AttrIBUTES.Email">

<font face="Arial" size="+1" color="Blue"><b>Export
Status</b></font><br>

<cfswitch expression="#AttrIBUTES.Delimiter#">
  <cfcase value="Comma">
    <cfset AttrIBUTES.Delimiter = ",">
  </cfcase>
  <cfcase value="Tab">
    <cfset AttrIBUTES.Delimiter = Chr(09)>
  </cfcase>
</cfswitch>

<!-- Query table & check for records to export -->
```

```
<!-- Get field list -->
<cfquery name="Field_List" datasource="Demo"
username="Guest" password="Guest">
    SELECT syscol umns. Name
    FROM sysobjects , syscol umns
    WHERE syscol umns. id = sysobjects. ID
    AND upper(sysobjects. Name) = '#Attributes. Table#'
</cfquery>

<!-- Put field list into an array -->
<cfset x = "">
<cfloop query="Field_List">
    <cfset x = x & Field_List. Name>
    <cfif Field_List. CurrentRow NEQ Field_List. RecordCount>
        <cfset x = x & Attributes. Delimiter>
    </cfif>
</cfloop>
```

```
<!-- Create export file -->
<cffile action="WRITE" file="#Attributes.FileName#" out-
put="" addnewline="No">
```

```
<!-- Check If user wants field names in file header -->
<cfif ucase(AttrIBUTES.ColumnHeader) EQ "YES">
<!-- Write out field names to the text file header-->
<cf file action="APPEND" file="#AttrIBUTES.FileName#" out-
put="#x##Chr(13)##Chr(10)#" addnewline="No">
</cfif>
```

```
<!-- Export the records -->
<!-- Build records -->
<cfset FL = ListToArray(x, #Attributes.Delimiter#)>
<cfloop query="Export">
  <cfset d = "">
  <cfloop index="n" from="1" to="#ArrayLen(FL)#">
    <cfset S = SetVariable("S", "#FL[n]#")>
    <cfset d = d & Trim(Evaluate(S))>
    <cfif n NEQ ArrayLen(FL)>
      <cfset d = d & Attributes.Delimiter>
    </cfif>
  </cfloop>
  <!-- Write record to text file -->
  <cffile action="APPEND" file="#Attributes.FileName#" out-
put="#d#&#Chr(13)##Chr(10)#" addnewline="no">
</cfloop>
```

```
<!-- Check if user wants export file emailed -->
<cfif Attributes.Email NEQ "">
  <cfmail to="#Form.Email#" from="Array_Table_Export" sub-
    ject="File Export" mimeattach="#Attributes.FileName#" serv-
    er="arrayone.com" port=25 timeout=600>
    File export completed. Rows = #Export.RecordCount#
    Here is a copy of the export file
  </cfmail>
  <cfoutput>The export file, #Attributes.FileName# was
    emailed to #Attributes.Email#. </cfoutput><br><br>
</cfif>
```

```
File export done.<br>
The new export file name is <cfoutput>
#Attributes.FileName#</cfoutput>. <br>
```

CODE  
LISTING

The code listing for  
this article can also be located at  
[www.ColdFusionJournal.com](http://www.ColdFusionJournal.com)

# Allaire

**[www.allaire.com](http://www.allaire.com)**



## A Beginner's Guide to ColdFusion

# COLDFUSION BASICS

Part 1



## A primer in dynamic page development

FROM THE BOOK  
BY BEN FORTA

### Using Templates

All ColdFusion interaction is via templates rather than HTML files. Templates can contain HTML, ColdFusion tags and functions, or both.

ColdFusion templates are plain text files, just like HTML files are. Unlike HTML files, which are sent to the user's browser, templates are first processed by ColdFusion. This allows you to embed instructions to ColdFusion within your templates. If, for example, you wanted to process user-passed parameters, retrieve data from a database, or conditionally display certain information, you could instruct ColdFusion to do so.

Instead of just reading about templates, why don't we create one?

The first template you create will just say hello to you. You can do that with any HTML file, but along with saying hello, this template also identi-

fies your IP address and the browser you are using. You can't do that with plain HTML.

Create a text file containing the code in Listing 1 and save it in your C:\A2Z\SCRIPTS\11 directory as HELLO1.CFM. (Note: The instructions here [and throughout this series of articles] assume that source files are saved in a directory structure named C:\A2Z\SCRIPTS, with a subdirectory for each month that an article appears in *ColdFusion Developer's Journal*. If you have not already done so, create a directory to store the files that you create as you work through these articles. You also need to create a Web server mapping [or alias] to map the virtual path a2z to the physical directory C:\A2Z\SCRIPTS. Refer to your Web server documentation for instructions on how to do this. If you are using Microsoft IIS or Personal Web Server, you must also ensure that the a2z alias has execute privileges.)

Once you have created and saved the file, load your browser and enter `http://yourserver.com/a2z/11/hello1.cfm` in the URL field (replacing yourserver.com with your own server name or IP address).

### TIP

If you are running ColdFusion on the same machine that you are developing on, you can use the address localhost to refer to your own computer. localhost is a special host name (which maps to the IP address 127.0.0.1) that always points to itself. In an environment where you are using dynamic IP address (dialing up to your ISP or using DHCP on a company network, for example) this is the only host name that always works, regardless of the IP address actually assigned.

Your browser will display a page that should look similar to the one shown in Figure 1; of course, your IP address and browser information could be different.

### Understanding ColdFusion Templates

Now take a look at the code in Listing 1. Most of the code should be familiar to you as standard HTML. The tags for head, title, line breaks

*This article has been adapted from Chapter 11 of ColdFusion 4 Web Application Construction Kit by Ben Forta. Published by permission of Macmillan Publishers Ltd. and the author. Part 2 of Chapter 11 will appear in the March issue of ColdFusion Developer's Journal, to be followed by adaptations of Chapters 12 and 13. The book can be purchased through Amazon.com or by clicking on [www.forta.com/books](http://www.forta.com/books).*

and bold text are the HTML that you'd use in any other Web page.

What is not standard HTML? The `<CFOUTOUT>` tag and fields surrounded by pound signs (#).

All ColdFusion-specific tags begin with CF, and `<CFOUTPUT>` is a ColdFusion-specific tag. `<CFOUTPUT>` (or ColdFusion output) is used to mark a block of code that ColdFusion should itself process prior to submitting it to the Web server for sending to your browser. When ColdFusion encounters a `<CFOUTPUT>` tag, it scans all the text until the next `</CFOUTPUT>` for ColdFusion functions or fields delimited by pound signs.

There are two fields used in Listing 1: `#REMOTE_ADDR#` and `#HTTP_USER_AGENT#`. They are CGI variables that HTTP servers make available to CGI applications like ColdFusion. `#REMOTE_ADDR#` contains your browser's IP address, and `#HTTP_USER_AGENT#` contains the string that your browser identified itself with. When ColdFusion encountered the text `#REMOTE_ADDR#` in the `CFOUTPUT` block, it replaced it with the value in the `REMOTE_ADDR` CGI variable. When it encountered `#HTTP_USER_AGENT#` on the next line, it replaced that with the appropriate CGI variable. Instead of sending the text you entered back to your browser, ColdFusion replaced the field names with the field values, and sent that back to you instead.

Why did we need the `<CFOUTPUT>` block? Take a look at what ColdFusion would have done without it. Listing 2 contains a modified version of the code used earlier, this time twice – once within a `CFOUPUT` block and once without.

If you use fields outside of a `CFOUTPUT` block, ColdFusion displays the field name as you entered it, complete with the delimiting characters. You can see this in Figure 2. More often than not, this is not the result you'll want.

## TIP

Every `<CFOUTPUT>` tag must have a corresponding `</CFOUTPUT>` tag, and vice versa. ColdFusion returns a syntax error if you omit either tag.

## Passing Parameters to Templates

In the first example you used ColdFusion to display dynamic data by specifying the field names for two CGI variables. ColdFusion can be used to display process parameters passed to a URL in exactly the same way.

To pass a parameter to a template, the parameter name and value are specified within the URL. For example, to pass a parameter `NAME` with a value of `BEN`, add `?NAME=BEN` to the URL. If you specify multiple URL parameters, each one must be separated by an ampersand character (&).

Try this yourself. Listing 3 contains a template that displays – if it exists – the value of a parameter called `NAME`. To do so, it uses the `<CFIF>` tag to create a condition, and a ColdFusion function called `IsDefined()`. If the parameter `NAME` exists, its value is displayed; otherwise, the user is notified that the parameter was not passed.

Once you have created and saved the file as `HELLO3.CFM` in the `C:\A2Z\SCRIPTS\11` directory, load your browser and type `http://your-server.com/a2z/11/hello3.cfm?NAME=BEN`. (You don't have to use my name, any name will do.) Your browser display should look like the one shown in Figure 3. Now try it again without any `NAME` parameter. This time you should see a display like the one shown in Figure 4.

Why did we bother testing for `IsDefined("name")`? Try removing the `<CFIF>` statement (you have to remove the `<CFELSE>` and `</CFIF>` lines too) and then enter `http://yourserver.com/a2z/11/hello3.cfm` without any `NAME` parameter. You'll see an error screen similar to the one shown in Figure 5. ColdFusion returns an error message because it has no idea what `#name#` is. You instructed ColdFusion to process a field that did not exist, and so it rightfully complained.

Now that you've seen what ColdFusion templates look like, and know how to create, save, and test them, return to A2Z Books.

Your employee database is set up and populated with data, and so your next task is to publish this information on your intranet. This way your users can use an up-to-date employee list at all times and won't need any special software to do so. All they need to access the data is a Web browser.

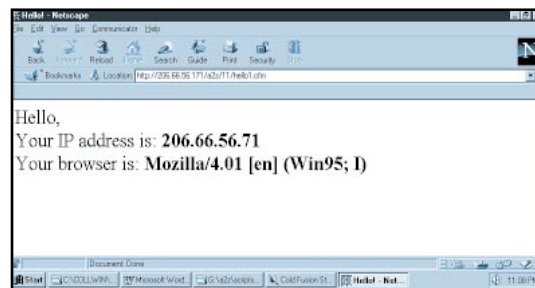


FIGURE 1 ColdFusion templates allow you to display dynamic data in your Web pages.

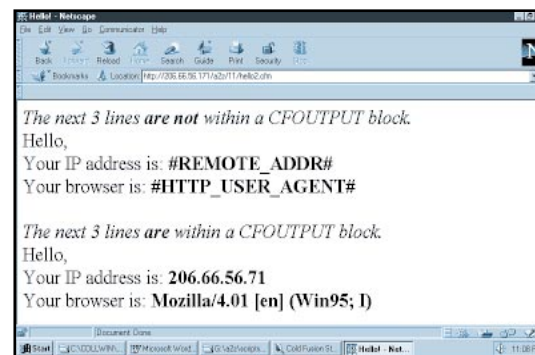


FIGURE 2 Fields not contained within a `CFOUTPUT` block will be output as is, and not replaced with their values.

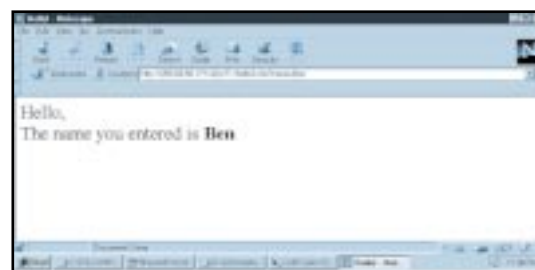


FIGURE 3 ColdFusion converts parameters passed to a URL into fields that you can use within your template.

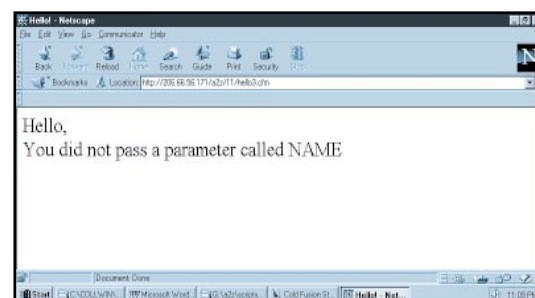
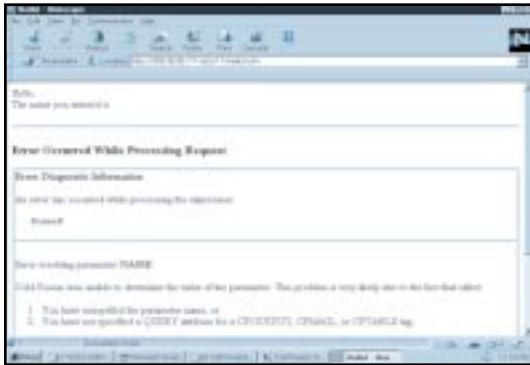


FIGURE 4 Whenever fields are optional, you should verify that they exist before using them.

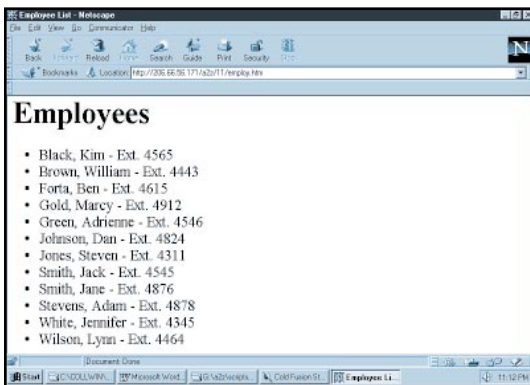
## Static Web Pages

Before we create the ColdFusion template, first take a look at how not to create this page.

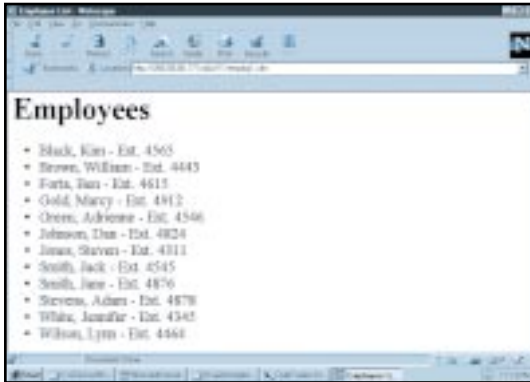
Listing 4 contains the HTML code for the employee list Web page. The HTML code is relatively simple; it contains header information and then a list of employees in an HTML unordered list (`<UL>`).



**FIGURE 5** ColdFusion displays an error message if you refer to a variable or field that does not exist.



**FIGURE 6** You can create the employee Web page as a static HTML file.



**FIGURE 7** Ideally, the employee Web page should be generated dynamically, based on live data.

Figure 6 shows the output that this code listing generates.

### Dynamic Web Pages

Why is a static HTML file not the way to create the Web page? What would you do when a new employee is hired, or when an employee leaves the company? What would you do if phone extensions changed?

You could directly modify the HTML code to reflect these changes, but you already have all this information in a database. Why would you want to have to enter it all again? You'd run the risk of making mistakes – names being mis-

spelled, entries out of order, and possibly missing names altogether. As the number of names in the list grows, so will the potential for errors to occur. In addition, employees will be looking at inaccurate information during the period between updating the table and updating the Web page.

A much easier and more reliable solution is to have the Web page display the contents of your Employee table; this way any table changes are immediately available to all employees. The Web page would be dynamically built based on the contents of the Employee table.

To create your first ColdFusion template, enter the code as it appears in Listing 5 and save it in the C:\A2Z\SCRIPTS\11 as EMPLOY1.CFM. (Don't worry if the ColdFusion code does not make much sense yet; I explain it in detail in just a moment.)

Now load your browser and type <http://yourserver.com/11/a2z/employ1.cfm> in the URL field (replacing yourserver.com with your own server name). The results are shown in Figure 7.

### Understanding Data-Driven Templates

Now compare Figure 6 to Figure 7. Can you see the difference between them? Look carefully.

Give up? The truth is that there is no difference at all. The screen shots are identical, and if you looked at the HTML source that generated Figure 7, you'd see that aside from lots of extra white space, the dynamically generated code is exactly the same as the static code you entered in Listing 4, and nothing like the dynamic code you entered in Listing 5.

How did the code in Listing 5 become the HTML source code that generated Figure 7? Review the code listing carefully.

### The CFQUERY Tag

The first lines in Listing 5 include a ColdFusion tag called `<CFQUERY>`, which submits any SQL statement to an ODBC data source. The SQL statement is usually a SQL `SELECT` statement, but could also be an `INSERT`, `UPDATE`, `DELETE`, a stored procedure call, or any other SQL statement.

*Note:* To follow these examples you must first create the A2Z data source. The simplest way to do this is by using the ColdFusion Administrator, as follows:

1. First, load the ColdFusion Administrator.

2. Click on the ODBC menu option.
3. To create a new data source, enter "A2Z" in the Data Source Name field, select Microsoft Access Driver as the ODBC Driver, then click the Add button.
4. Type the full path to the downloaded A2Z.MDB file in the Database File field, or click the Browser Server button to browse for it.
5. Click the Create button to create the data source. Assuming everything has worked, you'll see a success notification screen. If an error screen is displayed, make sure the specified path is correct and try again.

The `<CFQUERY>` tag has several attributes, or parameters, that are passed to it when used. The `<CFQUERY>` in Listing 5 uses only two attributes:

- The `NAME` attribute is used to name the query and any returned data.
- The `DATASOURCE` attribute contains the name of the ODBC data source to be used.

The query `NAME` we specified is `Employees`. This name will be used later when we process the results generated by the query. (*Note:* Query names passed to `<CFQUERY>` need not be unique to each query within your page. If you do reuse query names, subsequent `<CFQUERY>` calls will overwrite the results retrieved by the earlier query.)

We specified `A2Z` for the `DATASOURCE` attribute [name of the data source created in Chapter 9 of this book]. `SELECT FirstName, LastName, PhoneExtension FROM Employees ORDER BY LastName, FirstName` was the SQL statement used. This statement selects the columns we need from the Employee table and sorts them by last name plus first name.

### TIP

The SQL statement in Listing 5 is broken up over many lines to make the code more readable. While it is perfectly legal to write a long SQL statement that is wider than the width of your browser, these generally should be broken up over as many lines as needed.

# House Ad

**[www.sys-con.com](http://www.sys-con.com)**



When ColdFusion processes the template, the first item it finds is the ColdFusion tag `<CFQUERY>`. ColdFusion knows which tags it itself must process, and which it must pass to the server directly. `<CFQUERY>` is a ColdFusion tag, and therefore must be processed by ColdFusion.

When ColdFusion encounters a `<CFQUERY>` tag, it creates an ODBC request and submits it to the specified data source. The results, if there are any, are stored in a temporary buffer and are identified by the name specified in the NAME attribute. All this happens before ColdFusion processes the next line in the template.

The `<CFQUERY>` code, and indeed all ColdFusion markup code, never gets sent on to the server for transmission to the browser. Unlike HTML tags, which are browser instructions, CFML tags are ColdFusion instructions.

It is important to note that at this point no data has been displayed. `<CFQUERY>` retrieves data from a database table, but it does not display that data. Actually, it does nothing at all with the data – that's your job.

The next lines in the template are standard HTML tags, headers, title and headings. Because these are not ColdFusion tags, they are sent to the Web server and then on to the client browser.

### *Displaying Query Results with the CFOUTPUT Tag*

Next we create an HTML unordered list using the `<UL>` tag. The list is terminated a few lines later with a `</UL>` tag.

The list of employees itself goes between the `<UL>` and `</UL>` tags. Each name is a separate list item, and therefore begins with an HTML `<LI>` tag. Instead of listing the employees as shown in Figure 6, we used a `<CFOUTPUT>` tag.

`<CFOUTPUT>` is the same ColdFusion output tag we used earlier. This time, however, we are using it to create a code block that is used to output the results of a `<CFQUERY>`. In order for ColdFusion to know which query results to output, the query name is passed to `<CFOUTPUT>` in the QUERY attribute. The name provided is the same that was assigned to the `<CFQUERY>` tag's NAME attribute. In this case, the NAME is Employees.

The code between the `<CFOUTPUT QUERY="Employees">` and `</CFOUTPUT>` is the output code

block. ColdFusion uses this code once for every row that was retrieved. Because there are currently 12 rows in the Employee table, the `<CFOUTPUT>` code is looped through 12 times. And any HTML or CFML tags within that block are repeated as well, once for each row.

### *Using Table Columns*

As explained earlier, ColdFusion uses # to delimit fields. In addition to CGI variables and URL parameters, which we used at the beginning of this article, ColdFusion fields can also be columns retrieved by a `<CFQUERY>`. Whatever field is used, ColdFusion replaces the field name with the actual value. When ColdFusion processed the output block, it replaced `#LastName#` with the contents of the LastName column that was retrieved in the Employees query. Each time the output code block is used, that row's LastName value is inserted into the HTML code.

ColdFusion fields can be treated like any other text in an HTML document; any of the HTML formatting tags can be applied to them. In our example the query results need to be displayed in an unordered list. Each employee's name and phone extension is a list item, and is therefore preceded by the `<LI>` tag. Because the `<LI>` tag is included within the `CFOUTPUT` block, ColdFusion outputs it along with every row.

Look at the following line of code:

```
<LI> #LastName#, #FirstName# - Ext.
#PhoneExtension#
```

That code becomes the following for employee Kim Black at extension 4565:

```
<LI> Black, Kim - Ext. 4565.
```

Only the `<LI>` tag is within the `CFOUTPUT` block – not the `<UL>` and `</UL>` – because you want only one list, not many. If the `<UL>` and `</UL>` were within the `CFOUTPUT` block, you would have a new list created for each employee – definitely not the desired result at all.

Figure 7 shows the browser display that this template creates. It is exactly the same result as Figure 6, but without any new data entry whatsoever.

Welcome to ColdFusion, and the wonderful world of dynamic data-driven Web pages!

## Using Drill-Down Applications

The nature of the World Wide Web places certain restrictions on data interaction. Every time a Web browser makes a request, a connection is made to a Web server, and that connection is maintained only for as long as it takes to retrieve the Web page. Subsequent selections and Web requests create yet another connection; again, for the specific request.

Simple user interfaces that we take for granted in most commercial software, such as scrolling through previous or next records with the cursor keys, become quite complex within the constraints of Web pages and how they interact with Web servers.

One very elegant and popular form of Web-based data interaction is the drill-down approach. Drill down is designed to break up data and display only what is needed on a single page. Selecting an item in that page causes details about that item to be displayed. The process is called *drilling down* because you drill through the data layer by layer to find the information you need.

The employee page you just created, for example, displays a simple list of employees and extensions. What if you want to display more information such as title, department, and e-mail address? You could select more columns in `<CFQUERY>` and display them in the `<CFOUTPUT>` code, but that would clutter the screen and make it hard to use.

A better approach is to display less information on a page, and allow the user to click an employee's name in order to display more information about that employee. This approach – gradually digging deeper into a data set to find the information you want – is known as drilling down.

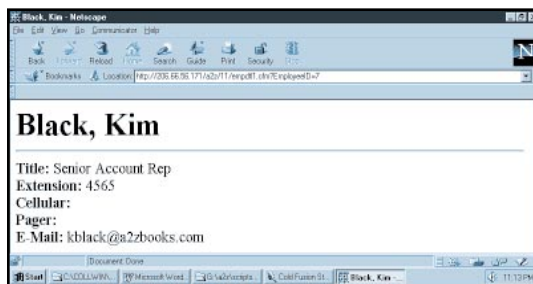
## Building Dynamic SQL Statements

Creating a drill-down application in ColdFusion involves creating multiple templates. In our example, one template lists the employees, and a second template displays an employee's details.

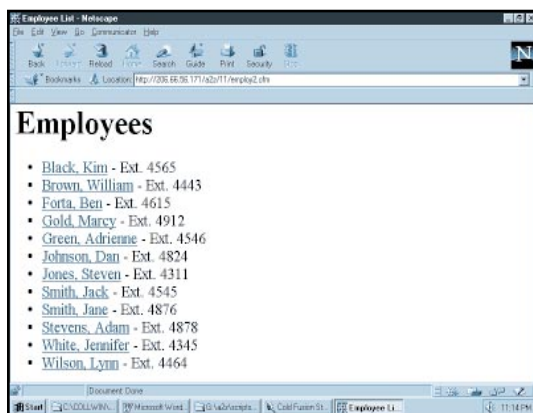
First create the detail template. The SQL query in this template has to select detailed user information for a specific user. Obviously you don't want to create a template for every employee in your database. Doing so would totally defeat the purpose of using templates in the first place. Rather, the template needs to be

# House Ad

**[www.sys-con.com](http://www.sys-con.com)**



**FIGURE 8:** If you want to create truly dynamic pages, parameters can be passed to ColdFusion templates and used to create dynamic SQL statements.



**FIGURE 9:** You can build hyperlink URLs dynamically to create even more dynamic Web pages.



**FIGURE 10:** By passing parameters to a ColdFusion template, you can use the same template to display different records and bypass requiring a different HTML page for each.

passed a parameter, a value that uniquely identifies an employee. Fortunately, when we created the Employee table, we created a column called EmployeeID, which contains a unique employee ID for each employee in the table. The code in Listing 6 demonstrates how this is done.

Before you look at the Web page produced by this code, take a look at the SQL statement in this `<CFQUERY>` tag.

The SQL SELECT statement selects the columns needed and uses a WHERE clause to specify which row to select. The WHERE clause cannot be hard-coded for any particular

employee ID and therefore uses a passed field: #EmployeeID#. The #EmployeeID# field is passed to the template as part of the URL.

Therefore, if an EmployeeID of 7 were passed with the URL, the WHERE clause (WHERE EmployeeID = #EmployeeID#) would become exactly what you need to select the correct row:

```
WHERE EmployeeID = 7
```

As seen earlier, parameters are passed to URLs after the template name, and each parameter is separated by an ampersand character. To specify employee ID 7, you'd add ?EmployeeID=7 to the URL.

Now try this out. Type the URL `http://yourserver.com/a2z/11/empdtl1.cfm?EmployeeID=7` in the URL field (replacing yourserver.com with your own server name) in your browser. The resulting output is shown in Figure 8.

To display the details for another employee, all you need to do is change the value passed to the URL EmployeeID parameter. Try replacing EmployeeID=7 with EmployeeID=5, which displays information on a different employee. The same template can now be used to display details for any employee in the database because the Web page is data driven.

### Implementing Data Drill-Down

To complete the drill-down application, you need to modify the employee list page to include links to the employee details page.

The code for the updated template is in Listing 7.

Listing 7 is the same as Listing 5, with two exceptions. First, you now need the EmployeeID value, and so the SQL SELECT statement in the `<CFQUERY>` has been changed to also include this column. Second, the employee's name in the `<CFOUTPUT>` code block has been modified so that it is a hyperlink to the employee detail page.

The new employee name code reads as follows:

```
<LI> <A
  HREF="empdtl1.cfm?EmployeeID=#EmployeeID#>#LastName#, #FirstName#</A>
  [c:ccc] - Ext. #PhoneExtension#
```

When ColdFusion processes employee 7, this line becomes the following:

```
<LI> <A HREF="empdtl1.cfm?EmployeeID=7">Black, Kim</A> - Ext. 4565.
```

This way, the URL needed for the hyperlink is also dynamic. The URL built for each employee will contain the correct employee ID, which can be passed to the employee detail template.

Now try this example. Type the URL `http://yourserver.com/a2z/11/employ2.cfm` (replacing yourserver.com with your own server name). Figure 9 shows what the output looks like. The only difference between this display and the one in Figure 7 is that now the employee names are hyperlinks. You can click any one of these links to display employee details, as seen in Figure 10.

### Using Frames to Implement Data Drill-Down

One problem with the drill-down templates just created is that every time you view an employee's details you have to click your browser's Back button to return to the employee list page. A more usable approach is to display the employee list and details at the same time.

Fortunately you can easily do this via a browser feature called *frames*. Frames allow you to split your browser window in two or more windows and control what gets displayed within each.

ColdFusion templates are very well suited for use within frames.

Creating frames involves creating multiple templates (or HTML pages). Each window in a frame typically displays a different template; you need two templates if you have two windows. In addition, there is always one more page that is used to lay out and create the frames.

When the frames are created, each window is titled with a unique name. In a nonframed window, the new page is opened in the same window every time you select a hyperlink, replacing whatever contents were there previously. In a framed window you can use the window name to control the destination for any output.

### Creating Frames for Use with ColdFusion

Now that you know how frames work, the first thing you need to do is create the template to define and create the frames. The code for template `EMPLFRAM.CFM` is shown in Listing 8.

**LISTING 1: HELLO1.CFM – Hello ColdFusion Template**

```

<HTML>

<HEAD>
<TITLE>Hello! </TITLE>
</HEAD>

<BODY>

<CFOUTPUT>

Hello, <BR>
Your IP address is: <B>#REMOTE_ADDR#</B><BR>
Your browser is: <B>#HTTP_USER_AGENT#</B><P>

</CFOUTPUT>

</BODY>

</HTML>

```

**LISTING 2: HELLO2.CFM – CFOUTPUT Use**

```

<HTML>

<HEAD>
<TITLE>Hello! </TITLE>
</HEAD>

<BODY>

<!--The next 3 lines <B>are not</B> within a CFOUTPUT
block. --><BR>
Hello, <BR>
Your IP address is: <B>#REMOTE_ADDR#</B><BR>
Your browser is: <B>#HTTP_USER_AGENT#</B><P>

<CFOUTPUT>

```

```

<!--The next 3 lines <B>are</B> within a CFOUTPUT block. --><BR>
Hello, <BR>
Your IP address is: <B>#REMOTE_ADDR#</B><BR>
Your browser is: <B>#HTTP_USER_AGENT#</B><P>

</CFOUTPUT>

</BODY>

</HTML>

```

**LISTING 3: HELLO3.CFM – Demonstration of URL Parameter Passing**

```

<HTML>

<HEAD>
<TITLE>Hello! </TITLE>
</HEAD>

<BODY>

Hello, <BR>

<CFIF IsDefined("name")>
  <CFOUTPUT>
    The name you entered is <B>#name#</B>
  </CFOUTPUT>
<CFELSE>
  You did not pass a parameter called NAME
</CFIF>

</BODY>

</HTML>

```

**LISTING 4: EMPLOY.HTM – HTML Code for Employee List**

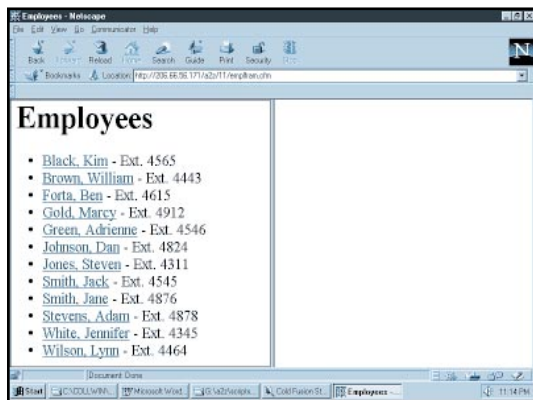
```

<HTML>
<HEAD>
<TITLE>Employee List</TITLE>

```

# Sitehosting.Net

**www.sitehosting.net**



**FIGURE 11:** ColdFusion is very well suited for use within HTML frames.

This template first defines the frames. `<FRAMESET COLS="50%,50%">` creates two columns (or windows), each taking up 50% of the width of the browser window.

The two columns are then defined: `<FRAME SRC="employ3.cfm" NAME="employees">` creates the left frame; the NAME attribute names the win-

dow; and the SRC attribute specifies the name of the template to initially display within the window when the frame is first displayed.

There is no employee selected when the frame is first displayed, and therefore there is no information to display in the details window (the right frame). You obviously can't display employee information in that frame before the user selects the employee to view, and so instead we display an empty page. SRC="blank.cfm" loads a blank page, the source for which is shown in Listing 9.

The next thing to do is create the employee list template. Actually, it is the same as the one in Listing 7, with one important difference. The URL to display the employee detail must include a TARGET attribute to designate which window to display the URL in. If the TARGET is omitted, the new data is displayed in the frame that it was selected from.

The modified code is shown in Listing 10. As you can see, the URL has been modified to include the attribute TARGET="details". This specifies that the new URL should be displayed in the frame we named Details, the right window.

That's all there is to it.

To try it out, go to <http://yourserver.com/a2z/scripts/11/emplfram.cfm>. Figure 11 shows the output as it appears in framed windows. Try clicking any employee's name in the left window; the right window will display employee details.



#### ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. In addition to authoring the book excerpted here, he is the author of its sequel, *Advanced ColdFusion 4 Development*. Ben also recently released *Sams Teach Yourself SQL in 10 Minutes*.

BEN@FORTA.COM

```
</HEAD>
<BODY>
<H1>Employees</H1>
<UL>
<LI>Black, Kim - Ext. 4565
<LI>Brown, William - Ext. 4443
<LI>Forta, Ben - Ext. 4615
<LI>Gold, Marcy - Ext. 4912
<LI>Green, Adrienne - Ext. 4546
<LI>Johnson, Dan - Ext. 4824
<LI>Jones, Steven - Ext. 4311
<LI>Smith, Jack - Ext. 4545
<LI>Smith, Jane - Ext. 4876
<LI>Stevens, Adam - Ext. 4878
<LI>White, Jennifer - Ext. 4345
<LI>Wilson, Lynn - Ext. 4464
</UL>
</BODY>
</HTML>
```

#### LISTING 5: EMPLOY1.CFM – The Employee List

```
<CFQUERY DATASOURCE="A2Z" NAME="Employees">
SELECT FirstName, LastName, PhoneExtension
FROM Employees
ORDER BY LastName, FirstName
</CFQUERY>

<HTML>

<HEAD>
<TITLE>Employee List</TITLE>
</HEAD>

<BODY>

<H1>Employees</H1>

<UL>

<CFOUTPUT QUERY="Employees">

<LI>#LastName#, #FirstName# - Ext. #PhoneExtension#
</CFOUTPUT>

</UL>

</BODY>
</HTML>
```

#### LISTING 6: EMPDTL1.CFM – Passing URL Parameters

```
<CFQUERY DATASOURCE="A2Z" NAME="Employee">
SELECT LastName,
       FirstName,
       MiddleInitial,
       Title,
       PhoneExtension,
       PhoneCellular,
       PhonePager,
       Email
FROM Employees
WHERE EmployeeID = #EmployeeID#
</CFQUERY>

<CFOUTPUT QUERY="Employee">

<HTML>

<HEAD>
<TITLE>#LastName#, #FirstName# #MiddleInitial#</TITLE>
</HEAD>

<BODY>

<H1>#LastName#, #FirstName#</H1>

<HR>

<B>Title:</B> #Title#<BR>
<B>Extension:</B> #PhoneExtension#<BR>
<B>Cellular:</B> #PhoneCellular#<BR>
<B>Pager:</B> #PhonePager#<BR>
<B>Email:</B> #Email#<BR>

</BODY>

</HTML>

</CFOUTPUT>
```

#### LISTING 7: EMPLOY2.CFM – ColdFusion Fields Can Be Used to Build

```
<CFQUERY DATASOURCE="A2Z" NAME="Employees">
SELECT FirstName, LastName, PhoneExtension, EmployeeID
FROM Employees
ORDER BY LastName, FirstName
</CFQUERY>
```





# Hidden Gems in 4.0.1

## What You Might Have Missed

BY  
CHARLES  
AREHART



Given that most people are now introducing and installing ColdFusion version 4.5, Allaire's latest release of the server and studio software, it may seem strange to read an article on 4.0.1.

Isn't 4.0.1 old news? And anyway, wasn't it just a maintenance fix?

There *were* new features introduced in the release. While the Allaire Release Notes and Web highlight pages mentioned several of the more prominent features, there were still more – far more – than you would expect in a mere maintenance release.

I've counted more than 60 new features or changes, and many of them aren't well documented, if at all. I'll discuss some of the most important ones here. I'll also provide links to the Allaire 4.0.1 documents.

### Why This Is an Important Subject, Even at the Dawn of 4.5

Even if you think you know what was new in 4.0.1, you owe it to yourself, your organization and your associates to review this article to be sure you haven't missed out on anything. Many of them are features or improvements that, if you don't know they're there, you might never be able to take advantage of them. They're not all that obvious.

For those who may be skipping from 4.0 to 4.5 because they just didn't notice 4.0.1, I hope this serves an even more vital role. If you only read about the new features in 4.5, you might miss out on the many important new ones that came in 4.0.1. In some instances you may encounter errors due to changes that were introduced in 4.0.1 but may not be documented as "changed" in the 4.5 release notes (since they're really not new to 4.5).

### More Than a Maintenance Fix

Maintenance releases are just supposed to fix things that are bro-

ken, and even in announcing the pending 4.5.1 release Allaire has said that it won't introduce new features.

Version 4.0.1, released as a free update to all 4.0 customers in April 1999, offered a virtual cavalcade of new features. Many of them were quite important, strategic, enterprise-level enhancements.

Here are a few that were highlighted in the Allaire release notes:

- Failover clustering support in NT
- Support for advanced security in Solaris
- New native database drivers for Informix and DB/2
- New UNIX support for HP/UX
- European currency support

The problem is that there are more than nine documents – if you can find them – on the Allaire site that describe all the changes. Even if you read the most significant documents, you could have missed dozens of other changes since no single document pulls it all together.

Many smaller features, not as well documented – what I consider truly hidden gems – introduce important new functionality that might thrill the day-to-day CF programmer or Studio user who missed them:

- Onrequestend.cfm, the corollary to application.cfm that executes at a template's conclusion
- Short-circuit evaluation where, for instance, you can now say `<CFIF is defined("form.size") and form.size is not "small">` without receiving an error when `form.size` doesn't exist
- A new ListQualify function that can take a list of string values and place quotes around them, which is particularly important

for use with a `SELECT ... IN` clause, for example

- A new PASSTHROUGH attribute for CFFORM, CFINPUT and CFSELECT that allows you to specify style sheet and other HTML attributes that Allaire hadn't thought to include in those tags
- Cached database connections, which can be released on demand or via an optional timeout (to allow updates to locked Access files without having to restart the server or play games with causing SQL error to unlock a file)
- A date/time stamp request feature\* that makes CFINSERT and CFUPDATE mark the date/time of the insert/update rather than the time the form was sent to the browser; it uses a hidden form field with the name of the intended column and a value of Currentdate()
- The ability to use an ODBC data source for advanced security authentication – a boon for those unable or unwilling to use an operating system security domain or LDAP server for authentication
- The ability to use Oracle and LDAP databases for storage of advanced security configuration

There are still more than 50 other new features in CF Server. And there are quite a few useful changes in CF Studio too, including but not limited to the following:

- Pressing F1 while the cursor is on a function or tag will open that function's Help page.
- The Help system's "find" feature now leverages a Verity index, making searches much faster.
- You can wrap selected text in

# **Career Opportunities**

pound signs using CTRL+3 (which makes sense, if you notice that 3 has the pound sign above it).

Some of the new features are changes in behavior. If you aren't aware of them, they can cause failures in applications that aren't modified. They're also important for those skipping from 4.0 to 4.5:

- CFAPPLICATION must now specify a NAME attribute.
- CFABORT no longer stops compilation/interpretation of a template, so if the code following it gets an error, it will now do so even though code isn't executed.
- If you use a session variable named "sessionid", you'll have conflicts with the new automatically created 4.0.1 session variable of the same name.

Allaire documented most of these changes and enhancements, but they were spread among several release notes and Web documents. I've scoured the Allaire site for information on this subject and have come across still more undocumented new features.

Speaking of documentation, there were new additions to the Allaire manuals as well as substantive changes. For example:

- *Using ColdFusion Studio* took a lot of information from the *Developing Web Applications* manual and also expanded on important Studio subjects.
- *A Quick Reference* serves as a handy listing of CF tags, functions and variables.
- The *Getting Started and Administering ColdFusion Server* manuals have been revised.
- The *Advanced ColdFusion Development* manual was discontinued and its contents folded into the other manuals.

### Just a Few More

There are so many other new features, but I'll just name a few more of the most significant ones.

- There is now an option to cause "stack trace" tracking in exception handling, controllable in the Administrator and with a new `cfcatch.tagcontext` variable.
- There is also an option to control the display of a template's path

on a CF error screen.

- You can now perform authentication using certificates.
- You can now "write to" Form and URL variables,\* setting them with CFSET and CFPARAM. The benefit: being able to create data for a CFINSERT/CFUPDATE on a form action page rather than solely on the form.
- A new "type=readonly" attribute was added to CFLOCK so you don't always have to perform an exclusive CFLOCK.
- The DateCompare now has a third "datepart" parameter to allow comparison on less than the complete date and time.
- ListValueCount counts the number of instances of a value in a list.
- StructKeyList lists the key names (elements) in a structure.
- Improvements were made to custom tags (passing structures) and exception handling (user-defined exceptions).
- There's a new "type" parm for CFPARAM (any, array, boolean, date, numeric, query, string, struct, UUID, variable name).
- There's a new IsProtected function.
- They've added support for x.509 certificates.
- Administrator pages no longer show debugging info, even if debugging is turned on.

The new release fixed a couple of rather severe security issues, including limiting the sample application "expression evaluator" to respond to page requests only from the machine on which it's installed.

### And in Studio?

There are several other new features in Studio 4.0.1, in addition to those mentioned above. For example:

- You can create, rename and delete folders in remote connections (both FTP and RDS).
- Pressing Ctrl-Shift double-click will select that tag, its end tag and elements within it.\*
- There's an option to turn on an outline bar around the current line selected (in options>settings>outline current line).
- You can specify that Netscape Navigator should be used as the internal browser for CF Studio

and HomeSite, if you have the experimental NGLayout (Gecko) engine from mozilla.org installed (probably not worth the trouble since then you'd be browsing with a version that your users likely wouldn't have. Use the "external browser" feature instead).

- There are new and improved tag editors.
- There's now a warning when you try to use Design mode on a cfm file.
- A remote connection to an FTP server can designate the remote directory as being relative to either the Web root or a server-specified user directory.
- There's support for the third-party link verification tool Linkbot 4.0.
- The Table Sizer (Quick Table) control on the Table tab can now expand to the limit of the screen.

### Sometimes It's the Little Things

One subtle change you might not notice will mean a lot to keyboard mavens: if during file>open and save dialogs you press the enter key while selecting a directory, Studio will now open that directory and display its files. In 4.0 it would attempt to open or save the file instead. Those who know the frustration of this nonstandard behavior have leaped (or will leap) at the discovery.

Sometimes it really is the little things!

Along the same lines, if you change the width of columns in the resource tab (such as the date/time or file size columns), they'll be saved when Studio is closed. It didn't do that previously, so many developers simply stopped trying to arrange these columns. There is also a new "view" option when you right-click on this area, to choose whether to even list the date modified, document type, size, and so on. Suffer no more!

Still more that may be trivial to some but a godsend to others: you can change the default behavior of File>Save As so that it (more logically) places its new file into the same directory from which the source file was opened. Otherwise, the default (to many, illogical) behavior is to store it instead in the directory currently pointed

to in the resource tab. The setting is in options>settings>general under the perhaps paradoxically named "display current local folder in file dialogs."\* By "current local folder" it means the one pointed to by the resource tab, not the "current" directory in which the source file was located. To clarify, you'll probably want to "deselect" that option to get what I've referred to as the "more logical" behavior. But some like the other approach, and so as not to confuse folks used to the "default" behavior, it remains an option that you can control and must change manually.

One last thing about Studio 4.0.1. It's not a benefit but rather an odd bug. Sometimes, for reasons that Allaire has not been able to pin down, you may find that on exiting the application you receive Windows "access violation" errors. This has been addressed with a replacement version discussed in Allaire Knowledge Base article 11868.

## Whither the Documentation

Where can you learn more? Well, of course, you can read the Allaire documentation, such as the *CFML Language Reference* and *Developing Web Applications in ColdFusion*. These certainly document the features but don't really identify *what* things have changed. For that you can find more in the 4.0.1 Release Notes and several Web documents that highlight many of the new features, such as:


1. 4.0.1 Update Summary:  
[www.allaire.com/handlers/index.cfm?ID=10719&method=full&LocationID=324](http://www.allaire.com/handlers/index.cfm?ID=10719&method=full&LocationID=324)
2. Server 4.0.1 Release Notes:  
[www.allaire.com/handlers/index.cfm?ID=10735&Method=Full](http://www.allaire.com/handlers/index.cfm?ID=10735&Method=Full)
3. Studio 4.0.1 Release Notes:  
[www.allaire.com/Handlers/index.cfm?ID=10025&Method=Full&Cache=Off](http://www.allaire.com/Handlers/index.cfm?ID=10025&Method=Full&Cache=Off) and  
[www.allaire.com/handlers/index.cfm?ID=10734&Method=Full](http://www.allaire.com/handlers/index.cfm?ID=10734&Method=Full) (seems a duplicate)
4. Studio 4.0.1 Update FAQ:  
[www.allaire.com/products/cold-](http://www.allaire.com/products/cold-fusion/faqs/401updatefaq.cfm)

[fusion/faqs/401updatefaq.cfm](http://fusion/faqs/401updatefaq.cfm)

5. Documentation Updates for ColdFusion 4.0.1:

[www.allaire.com/handlers/index.cfm?ID=10429&method=full](http://www.allaire.com/handlers/index.cfm?ID=10429&method=full)

## There You Have It!

As I hope these lists have demonstrated, 4.0.1 was much more than just a maintenance release. There were so many new features that ranged from the relatively minor to some fairly significant ones. Again, it's really important that you become familiar with most if not all of them. They'll almost certainly make you more productive, leaving more time for...learning about 4.5. 

\* Testing revealed that four of the items listed in the 4.0.1 documents as new were in fact present in 4.0. Perhaps they had not been documented in 4.0. Even so, they are interesting and easily missed, so we have kept them here.

## ABOUT THE AUTHOR

Charles Arehart is an Allaire-certified trainer and developer working with Fig Leaf Software and is a frequent speaker at user groups throughout the country.

CAREHART@FIGLEAF.COM

# Enhanced Technologies

[www.enhtech.com](http://www.enhtech.com)

**THE POWER OF E-COMMERCE IN ONE LITTLE CARD**

In October 1999, Enterprise USA Inc., makers of PerVirtualLinkpoint, proudly announced the release of CardFusion™ for use with Allaire's ColdFusion 4.x. CardFusion™ is a revolutionary software product that dramatically reduces development time of live credit card processing e-Commerce web sites. Execute live credit card transactions over the Internet from a single, easy to use CardFusion™ tag.

CardFusion is a custom CardFusion™ tag, complete with a tag wizard, a dialog based editor, and online help with numerous code examples. It allows merchants to use simple HTML and a single CardFusion™ tag to execute live credit card transactions via the Internet. Use CardFusion™ to integrate transaction processing into sophisticated data collection applications, customer databases, accounting, sales and marketing databases, or real-time executive decision support applications.

Enterprise USA Inc.  
248-888-1473  
Info@enterpriseusa.com  
[www.enterpriseusa.com](http://www.enterpriseusa.com)



# An Introduction to SQL

REVIEWED  
BY  
EMILY  
KIM

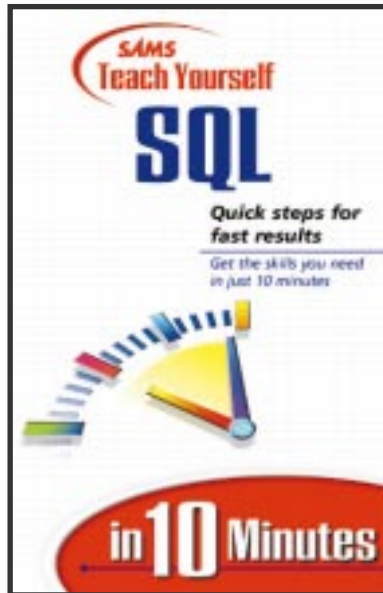


Who else is sick and tired of hearing Allaire's product evangelist, Ben Forta, touted as the CF god? Ben Forta this, Ben Forta that. All right already! So he and some other CF gurus wrote a couple books about CF and now we can't go anywhere in CF hyperspace without hearing about it.

Well, bad news for us. He's done it again. But this time in the related but separate arena of Structured Query Language (SQL). His *Teach Yourself SQL in 10 Minutes* is an excellent introductory book that also serves quite nicely as a pocket reference.

Before I go any further with this review and sing his praises, I have to disclose that I've worked with Ben on other projects. But don't think this makes me go goo-goo and starry-eyed in his presence. Quite the contrary – I'm actually one of his harshest critics. When I first picked up this book, it was with red pen in hand, ready as always to mark it up with corrections and comments. Unfortunately for my happy pen, this book is technically sound with only a few minor errors and the content is very well written. (In the near future you'll be able to find information about errors at [www.forta.com/books/0672316641](http://www.forta.com/books/0672316641).)

Bad news first: this book is *not* designed as an advanced discussion of SQL. I was really looking forward to a book that could take my SQL knowledge to the next level. However, while Ben certainly touches on some advanced topics, such as stored procedures and triggers, people who already know SQL well or who've worked with it on a daily basis will probably find that it's more useful as a pocket reference rather than a learning tool.



Nevertheless, I still recommend that nonbeginners who don't have formal training with SQL read the book because, while the majority of it will probably be review, you may be able to fill in some gaps in your SQL knowledge. There are plenty of statements littered throughout the book that clarify SQL syntax and a fair number of gold nugget hints about best practices and SQL optimization.

This book also abounds with *Notes, Tips* and *Cautions* that help clarify points and try to keep you out of trouble. In one tip dealing with transaction processing, Ben states: "You cannot roll back CREATE or DROP operations. These statements may be used in a transaction block, but if you perform a rollback they will not be undone." While this may seem intuitive to some, it wasn't obvious to me and I was glad to see it emphasized.

The book begins with a great overview of SQL but doesn't become too lecture-driven. There's at least one example SQL statement on almost every page of the book, and associated scripts and database files

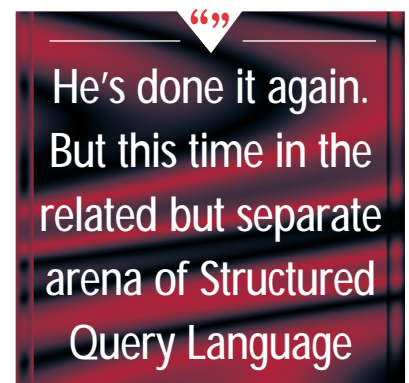
## Sams Teach Yourself SQL in 10 Minutes

By Ben Forta  
208 pages, Macmillan

that can be downloaded from the Web site (the URL listed earlier in this review). This allows those over-achievers to try out each line of code for themselves. And for those of you who have to work with different RDBMSs, Ben also tries to point out syntax differences between SQL Server and Oracle when either differs from standard SQL.

Last but not least, the appendices (unlike many other books) also have some useful material. Besides the review of SQL data types, there's a list of frequently used statements and – my favorite – a list of SQL reserved words.

Try as I might, I found it difficult to be discontented with *Teach Yourself SQL in 10 Minutes*. It packs an extraordinary amount of informa-



tion into a tiny package. Weighing in at 208 pages and about the size of half a piece of paper, it's amazing how much good, useful information it actually holds. While I'm still itching for more advanced, in-depth examples, I'm glad to have the book sitting on my reference shelf.

EMILY@TRILEMETRY.COM

**ABOUT THE AUTHOR**  
Emily Kim, director of development solutions and cofounder of Trilemetry, Inc., is a certified Allaire instructor. She serves as a technical and developmental editor and technical reviewer on computer books for several publishers and is a contributing author to *Mastering ColdFusion 4*.

# **Career Opportunities**

# ColdFusion & Generator Stock Charts

## Creating Stock Chart graphs

BY  
ANDREW  
STOPFORD



Since my first article on Generator in *CFDJ* (Vol. 1, issue 5), Macromedia has released version 2. With this version you have full Flash 4 functionality and a host of new Generator objects to use, from tickers to tables. What I'll show in this article is the combination of Flash and another new object, Stock Chart.

### Stock Chart Object

The Stock Chart object will give us a stock chart graph just like those you see on the big commercial Web sites that track stock data. This object gives us several options, such as "candle" graphs and display and color options. I'll cover these in detail later in the article.

### Creating the Stock Object in Flash

Open up Flash (Generator objects are now freely available on the Macromedia Web site) and choose Window - Generator Objects from the menu. Here you'll be presented with Generator objects all lined up in a toolbar. What interests us is the Stock Chart object.

Drag the object onto your movie and the object's properties menu will open up.

### Stock Chart Data Source

The format for Generator objects changes from object to object. The Stock Chart object's data source has the following format.

open,	close,	high,	low,	HLABEL
34,	38,	40,	31,	Monday
38,	45,	48,	37,	Tuesday
45,	40,	45,	36,	Wednesday
40,	32,	42,	27,	Thursday
32,	36,	42,	29,	Friday

Each value represents a value for a given day - for example, for Monday its opening and closing values and its high and low values, data that's required for stock chart value for a given day. To show this off I'll first demonstrate how it's used with a text file.

### Using a Text File with a Stock Chart Object

First, create the text file, then modify the data source of the Stock Chart object to point to it. To do this, make sure that your object's properties window is open (double-click on the Stock Chart object in your movie) and open up the datasource window by clicking the little button in the datasource property.

Next, type the path to the data text file (i.e., C:\myfiles\data.txt) and click OK (if you've saved the Flash File in the same directory as the text file, you don't need to type the path, i.e., data.txt).

### Testing the Movie in Flash

Before we can run our Generator file, we first have to set up Flash. Open the Publish Settings box by selecting File - Publish Settings. Make sure that Generator is selected before you press OK.

To run our movie, choose Control - Test Menu from the menu options and your movie will look like Figure 1.

To make the chart a little easier to read, change the following values in the movie's properties.

Chart Property	Value
Chart Type	CandleSticks
Value Placement	Over Chart
Value Display	RollOver

Your chart will now look like the chart shown in Figure 2.

### Using ColdFusion with the Stock Chart Object

We can now replace our text file data source with a ColdFusion data

source. To do so I'll create a database data source and use ColdFusion to read the data and format it for Generator to use.

### Database Data Source

I created the data source in an Access Database as shown in Figure 3.

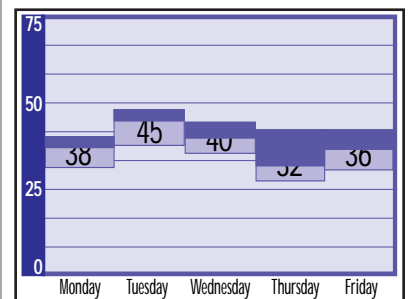


FIGURE 1: Standard Chart output

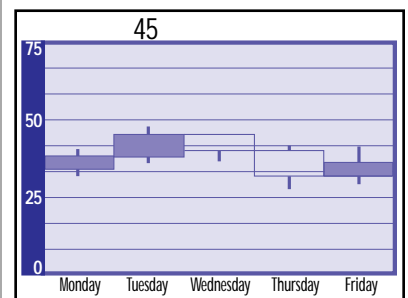


FIGURE 2: Modified Chart output

### The CFM Code

The next step is use ColdFusion code to read the database and format the data in a format that Generator will understand; we're really trying to emulate the text file data source that we used earlier.

```
<cfquery name="stock1" data
source="kimmuli" dbtype="ODBC">
```

```
SELECT open, close, high, low, HLABEL
```

	open	close	high	low	HLABEL
▶	34	38	40	31	Monday
	38	45	48	37	Tuesday
	45	40	45	36	Wednesday
	40	32	42	27	Thursday
	32	36	42	29	Friday
*					

FIGURE 3 Database data

```
FROM      stock
</cfquery>
<cfcontent
type="text/plain">open, close,
      high, low, HLABEL
<cfoutput query="stock1">#open#,
#close#, #high#, #low#, #HLABEL#
</cfoutput>
```

The code reads the database and displays it in a text format. Note the lack of HTML code in the code. This is because the data source can't con-

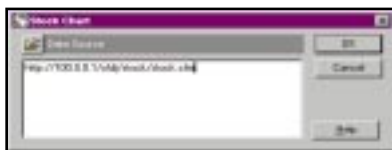


FIGURE 4 The ColdFusion URL page in datasource window

tain characters other than those needed to be present in the data source (HTML characters would cause errors). If you do need other characters, it's possible to use escape codes. See the Macromedia Generator support site technotes for details.

### Adding the CFM File to the Data Source

The final step is to add the CFM page to the Generator data source of our Stock Object, as shown in Figure 4.

Note that you must use the full HTTP path to the CFM file. Once you've added the file, clicked OK and tested it, you should see something like Figure 5.

A nice feature of Flash is the

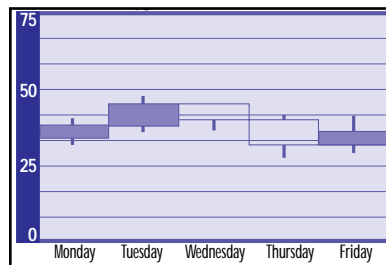


FIGURE 5 Chart output from ColdFusion page

ability to test out all the elements of your Generator object before you test them on your Web site. Remember to change the formatting of the Stock Object as I did, using the text example at the start of this article.

### Summary

Generator 2 has many new features, and one of the most visually appealing is the Stock Chart. Here I've shown how you could use either a text file or a Database/CFM page to get the data to the Stock Chart.



ANDREW.STOPFORD@VIRGIN.NET

### ABOUT THE AUTHOR

Andrew Stopford is a Web developer and consultant from south Manchester in the UK and a Macromedia evangelist for Generator. He has lent his hand to many Generator sites around the world. Andrew's kept busy answering questions that the Generator community posts in the Macromedia Generator NG and other information sites. He's also the creator of Kimmuli, a code tool for Generator.

# ISite Design

www.isitedesign.com

**BUYSPECTRA.COM**

Your source for  
the entire family  
of Allaire products

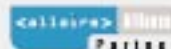
<Owned and operated by Allaire Spectra® enthusiasts>

Save up to 50% before March 1  
on all product purchases

<Free test drive of Allaire Spectra>

RFP builder to jumpstart your project with  
Allaire Spectra and ColdFusion® consultants

Call Rene today: 1.888.269.9103 ext. 107



BuySpectra.com is a  
CFcommunity LLC property

ColdFusion is a U.S. registered trademark, and Allaire, the Allaire logo, Allaire Spectra, and the Allaire Spectra logo are trademarks of Allaire Corporation. All other company names, brand names, and product names are trademarks of their respective holders. ©1999 Allaire Corporation.

# A Fusebox How-To

## Using Fusebox to gather requirements

BY  
HAL  
HELMS



I recently received an inquiry from a developer about my Guru-on-Call service ([www.TeamAllaire.com/hal](http://www.TeamAllaire.com/hal)). He requested help in identifying the fuses he would need for his application.

After reading my first two Fusebox articles in last year's *CFDJ* (Vol. 1, issues 3 and 4), he wrote, "I understand what you are explaining but implementing it is a little harder than I thought."

I wonder if others may be having this same problem – understanding the theory but a little hung up on the details. In this article I'm going to walk through developing a Fusebox application so you can see the theory put into reality. I'm assuming you've already read enough about the Fusebox methodology to understand how it works. If you need a refresher on this, check out the back issues of *CFDJ* online for the two articles I wrote with Steve Nelson and Gabe Roffman, or see the sidebar for a quick recap of Fusebox rules.

I find the hardest part of a large Web application is not the actual coding – thanks to Fusebox – but

determining what the actual job requirements are. I'm not alone in this; it's a common complaint of developers. Over the years I've tried, in a variety of ways, to nail down customers so they wouldn't keep changing their minds. One day it occurred to me that the reason customers *wouldn't* stop making changes is because they *couldn't*. They couldn't tell me the "requirements" until they saw them reflected in the application.

If this is true, it's my job to make sure the users can see the application – and prod, poke and punch it – until they're sure that what they see is what they want. Only then do I begin the actual coding. I place great emphasis on a simulated application that looks like the real thing to the users. When they click buttons, things happen. Links are live. Forms accept inputs. And while it looks like a real application, it's a

completely different matter under the hood. There's no database hooked up to the application. There are no persistent variables, perhaps no variables at all. Any use of code (CFML or otherwise) is there only to present a convincing simulation. We haven't begun coding yet; we're doing this because experience has shown that it's the only way to find out what kind of application we should build.

This methodology takes some of the stress off skilled programmers. In practice I find that over half the work required in developing an application can be done without involving programmers – a welcome discovery as they're hard to find. This prototype is handled by people skilled in interface design and graphical arts who create essentially static Web pages that mimic their real counterparts.

It's during this process that questions, comments and concerns surface. I needed a way to capture and contain this information in a central location where all those involved in the development of the application could communicate and contribute. Over time we've developed a method that effectively lets us define and refine what the application should be, do and look like.

This method relies on some ColdFusion code running alongside the designer's prototype work. Since designers typically aren't coders, we ask only that they save the Web files with a .cfm extension and append the following code onto each of these pages:

```
<cfinclude
template="devnotes/index.cfm">
```

Use Case	Fuseaction(s)	Fuse(s)
User wants to add a note within a category and associate that note with the page user is on.	addNote	dspPageNotes.cfm provides the user with a form for entering a new note. actInsertNote.cfm does the actual DB processing.
User wants to view notes previously entered for that page.	showPageNotes	dspPageNotes.cfm
User wants to see a listing of all notes.	showAllNotes	dspAllNotes.cfm
User wants to be able to delete a note.	deleteNote	dspPageNotes.cfm provides the user with a "delete" link. actDeleteNote.cfm does the actual DB processing.
User wants to be able to mark a note to change its status. This can be used by clients to identify questions they have answered, or by developers to show tasks that have been completed, etc.	setNoteStatus	dspPageNotes.cfm provides the user with a checkbox next to each note. actSetNoteStatus.cfm does the actual DB processing.

TABLE 1: Steps in Fusebox methodology



At runtime this code calls a small Fusebox application at the bottom of the page that produces something similar to Figure 1.

This allows us to preserve the history of the development of the application while providing a map for its continued evolution. I'm going to show you how I approached building this Fusebox-based mini-app, which I hope will make the process of creating a Fusebox application clearer than a mere reading of the rules.

I start off application development – even of small apps – by creating “use cases” to identify requirements for the application. These are natural language statements that require no technical background and are ideal for communicating between client and developer. At this point, use cases form the basis of our understanding of what the application should do. A sample use case might look like this:

*“User should be able to log onto the system and be validated as either a user or administrator.”*

While use cases are wonderful for determining requirements for the application and for communicating with clients, they're too general to be of much use to developers. For this I rely on the skill of interface designers who understand how to translate the client's requirements into actual pages that show how they'll do it. This is the prototype I spoke of above. It's an iterative process; at each step we hopefully come closer to finding exactly what the client needs. Once all participants have agreed that the application is fully defined, we arrive at a prototype freeze. Now it's my job to match the use cases (as interpreted in the prototype) with one or more fuseactions.

Fuseactions, remember, define what the application is actively involved in; at any point there's only one fuseaction operating. A fuseaction is a request for action that's sent only to the fusebox (usually named `index.cfm`). It's fundamental to Fusebox that all requests for action go through the fusebox, not to individual fuses. Without this we're on a slippery slope where one

Fusebox Rules

1. The application has a central page called a fusebox.
2. The fusebox is responsible for handling fuseactions that are passed to it. These are requests for some action such as logging in a user, displaying a menu or updating a database.
3. Fuses are small bits of code that are called on by the fusebox to perform some action in the course of responding to a fuseaction. Examples are “`dspShowUserLoginScreen.cfm`” and “`act-ValidateUser.cfm`”.
4. There's only one active fuse-action at a time.
5. Fuses define their exit states as members of a structure called “RFA” (for “return fuseactions”).
6. A fuse can never call another fuse directly, but must always return to the fusebox if another action is needed.
7. The Fusedoc section of a fuse provides the information needed to interact with that fuse.

fuse calls another and that fuse calls yet another until we end up with a tangled mess of intricate dependencies between fuses, defeating our goals of readability and reusability.

Once the request for action (the fuseaction) is sent to the fusebox, the fusebox calls on fuses to carry out the action. So the development process goes like this: use case → prototype → fuseaction(s) → fuse(s). By creating a table that shows the associations between use cases, fuses and fuseactions, I can see how complete my application architecture is. Table 1 is the table for this application.

As Dennis Miller says, “I don't want to get off on a rant here...,” but let me say a word about naming fuses. The Fusebox.org site suggests naming prefixes for fuses based on the fuse's job: `dsp_fuse`-name for display-type fuses and so forth. I've heard heated debate over the “correct” naming scheme and I think this misses the key point – Fusebox is a development methodology, not a naming convention. My position is, if you find the naming scheme to be helpful, by all means use it. If you have another naming scheme – possibly already a standard within your company – then use that. The power of Fusebox doesn't depend on how a fuse is named, but on its clarity, conciseness and precision.

Now that I know the fuses I'll

need, I can add Fusedoc comments (see my column in the last issue of **CFDJ** [Vol. 2, issue 1] for more on this) and a generic message to create a “fuse stub” for each fuse. See Listing 1 for my fuse stub for the fuse “`actDeleteNote.cfm`”.

This is such a small fuse (here, at least, smaller is better) that the line tends to blur between fuse stub and fuse. If you're just starting out writing fuse stubs, you may not be sure how far you should go. When does the stub become the actual fuse?

My test for fuse stub completeness is this: Can another coder who's competent in ColdFusion, but doesn't know the scope of the application, successfully complete the fuse? When the answer is Yes, the fuse stub is ready. If this were a display-type fuse, you'd see the

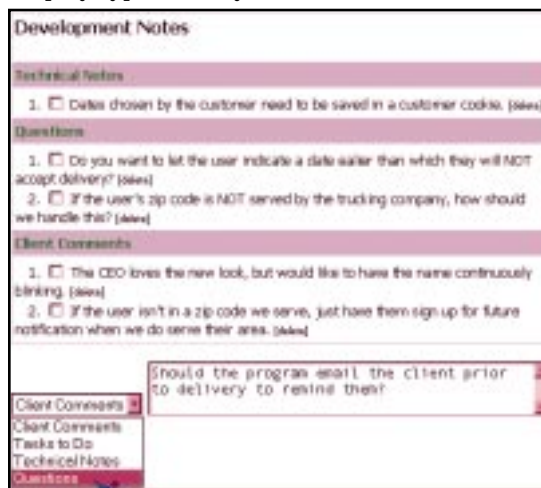


FIGURE 1 Screenshot of the applet in use

**Take a look  
at our specials  
this month!**

### EASTLAND DATA SYTEMS Internet Shopping with Java Shopping Cart

...Described as the most progressive and interactive form of shopping on the web today... This Java Applet provides a complete user interface package for Internet Shopping Web Sites. Using Java technology we produce a drag-and-drop shopping user interface that is fun and easy to use, encouraging shoppers instead of frustrating them with confusing controls that are hard to follow. And the easier it is to shop, the more you sell.



**\$294<sup>99</sup>**

### Hybrid Shopping Cart

This Java Applet provides a complete user interface package for Internet Shopping Web Sites. A "Hybrid" is defined as an offspring of two varieties. A blending of the best features from our CGI and Java shopping products, we took the most powerful aspects of Java technology: real-time, on-screen updating and computational capabilities. And combined those with the most desirable features of our CGI shopping Cart, namely it's flexibility and compatibility with web designers with artistic talent.



**\$294<sup>99</sup>**

### CGI Shopping Cart

The Shopping Cart automates the Shopping Process to make shopping on your site intuitive, straight forward, and enjoyable! It's one of the most affordable Shopping Carts because it was designed for small businesses. Specifically for entrepreneurs who are testing the Internet waters, and can't or don't want to make large investments into bells and whistles for their site. But simply want to make shopping on their site easy for the customer.



**\$294<sup>99</sup>**

### Guaranteed Best Prices

JDJ Store Guarantees the Best Prices. If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.

Terms of offer:

- Offer good through October 31, 1999
- Only applicable to pricing on current versions of software
- August issue prices only
- Offer does not apply towards errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.  
Not responsible for typographical errors.

**Attention Java Vendors:**  
To include your product in JDJStore.com,  
please contact jackie@sys-con.com



**GUARANTEED  
BEST PRICES  
FOR ALL YOUR  
JAVA RELATED  
SOFTWARE  
NEEDS**

ALLAIRE

### ColdFusion 4.0

With ColdFusion 4.0, create Web applications for self-service HR solutions, online stores, interactive publishing and much more. It's the integrated development environment that has all the visual tools you need to create Web applications quickly and easily. From simple to sophisticated ColdFusion gives you the power to deliver the Web solutions you need - faster, and at a lower cost.



**ColdFusion Studio 4.0** ..... **\$354<sup>99</sup>**  
**SkillBuilding with ColdFusion Interactive Training CD** ..... **\$284<sup>99</sup>**

ALLAIRE

### JRun - Live Software

JRun is the industry-leading tool for deploying server-side Java. JRun is an easy-to-use Web server "plugin" that allows you to deploy Java Servlets and JavaServer Pages. Servlets form the foundation for sophisticated server-side application development. Java servlets are platform independent, easy to develop, fast to deploy, and cost-effective to maintain.



**JRun** ..... **\$558<sup>99</sup>**

PROTOVIEW

### Java Enterprise Editions

The Java Enterprise Editions offer you a choice between comprehensive packages of award-winning AWT or JFC components along with enterprise-level support and subscription service. Powerful, extendable, lightweight components built on the foundation of JFC, the ProtoView JFCSuite contains JFCDataCalendar, JFCDataExplorer and JFCDataInput. The JSuite (AWT) includes the DataTableJ grid component with JDBC and Visual Café database support. Also includes TreeViewJ, CalendarJ, TabJ and WinJ.

**JSuite Enterprise Edition** ..... **\$818<sup>99</sup>**  
**JFC Enterprise Edition** ..... **\$818<sup>95</sup>**  
**JSuite** ..... **\$328<sup>99</sup>**  
**JFCSuite** ..... **\$408<sup>99</sup>**

GALILEO DEVELOPMENT SYSTEMS

### Intr@Vision Foundation

Intr@Vision Foundation helps bring ColdFusion development to the next level. It provides an out-of-the-box application security architecture for handling your most complex intranet and extranet needs. Instead of spending 30% of your development time adding security to every application you build, it gives you a proven solution with a single line of code. Intr@Vision Foundation allows your developers to focus on building business solutions, not infrastructure.



**Intr@Vision Foundation** ..... **\$3499<sup>99</sup>**

ALLAIRE

### HomeSite 4.0

HomeSite is the award-winning HTML editing tool that lets you build great Web sites in less time, while maintaining Pure HTML. Unlike WYSIWYG authoring tools, HomeSite gives you precise layout control, total design flexibility and full access to the latest Web technologies, such as DHTML, SMIL, Cascading Style Sheets and JavaScript. HomeSite 4.0 is the only HTML editor featuring a visual development environment that preserves code integrity.



**HomeSite 4.0** ..... **\$87<sup>99</sup>**

INSTALLSHIELD

### InstallShield Java Edition 2.5

InstallShield Java Edition 2.5 is the powerful tool developers require to produce bulletproof InstallShield installations with Java versatility. You can target your application for multiple systems with cross-platform distribution. And InstallShield Java Edition 2.5 offers the key features and functionality designed to let developers go further in distribution and deployment.



**InstallShield Java Edition** ..... **\$474<sup>99</sup>**

KL GROUP

### JProbe Suite

JProbe Profiler is the most powerful tool available for finding and eliminating performance bottlenecks in your Java code. JProbe Coverage makes it easy to locate individual lines of untested code and reports exactly how much of your Java code has been tested. JProbe Threadalyzer lets you pinpoint the cause of stalls and deadlocks in your Java applications and makes it easy to predict race conditions that can corrupt application data.



**JProbe Profiler w/ Standard Support (inc. JProbe Memory Debugger)** ..... **\$464<sup>99</sup>**  
**JProbe Coverage w/ Standard Support** ..... **\$464<sup>99</sup>**  
**JProbe Threadalyzer w/ Standard Support** ..... **\$464<sup>99</sup>**  
**JProbe Suite w/ Standard Support** ..... **\$934<sup>99</sup>**

CATOUZER

### Synergy SOHO

Synergy is a ColdFusion-based Web application framework for instantly deployable corporate intranets. It consists of an Application Services Layer, which offers central security and administrative services, and eight core collaborative applications. Synergy's open architecture is designed for implementing existing ColdFusion applications and developing new ColdFusion applications specifically tailored to the customer's needs.



**Synergy SOHO** ..... **\$499<sup>99</sup>**

**ORDER ONLINE**

**WWW.JDJSTORE.COM**

Once all the fuse stubs have been written, I can either parse these out among my team of programmers or I can write them myself. In both cases a good portion of the great time-waster – uncertainty – has been put to rest.

Right now there are no <CFCASE> statements – meaning this fusebox can't handle any fuseactions. But

If I've done a good job writing the fuse stubs, I can tell how close we are to completing the project by

I encourage you to look at the complete code files for this little application at [www.TeamAllaire.com/hal](http://www.TeamAllaire.com/hal). In addition to being a handy app, these code files show how I approach building a Fusebox application. And while you may find that you build your Fusebox applications differently, we can all benefit from having certain standards that we agree on.

*Hal Helms is a Team Allaire member living in Atlanta, Georgia. A frequent writer on ColdFusion and Fusebox, he also offers training and mentoring on these subjects.*

we agree on. 

The code listing for  
this article can also be located at  
[www.ColdFusionJournal.com](http://www.ColdFusionJournal.com)



# ADVERTISING INDEX

ADVERTISER	URL	PH	PG
ABLE SOLUTIONS	WWW.ABLECOMMERCE.COM	360.253.4142	2
ADHOST	WWW.ADHOST.COM	888 ADHOST-1	26
ALLAIRE	WWW.ALLAIRE.COM	888.939.2545	19,23,25,29
BIZNIZ WEB	WWW.WEBPUBLISHINGTOOLS.COM	281.367.4016	52
CAREER OPPORTUNITIES			41,45
CATOUZER	WWW.CATOUZER.COM	604.662.7551	55
DATARETURN	WWW.DATARETURN.COM	800.767.1514	4
DIGITAL NATION	WWW.DEDICATEDSERVER.COM	703.642.2800	3
EKTRON	WWW.EKTRON.COM	603.594.0249	9
ENHANCED TECHNOLOGIES	WWW.ENHTECH.COM	800.368.3249	43
ENTERACT	WWW.ENTERACT.COM	312.955.3000	11
ENTERPRISE USA	WWW.ENTERPRISEUSA.COM	248.888.1473	43
EPRISE	WWW.EPRISE.COM	800.274.2814	15
INFOBOARD	WWW.INFOBOARD.COM	800.514.2297	53
INTERMEDIA	WWW.INTERMEDIA.NET	650.424.9935	56
ISITE DESIGN	WWW.ISITEDESIGN.COM	888.269.9103	47
ON-LINE DATA SOLUTIONS	WWW.COOLFUSION.COM	516.737.4668	54
RSW SOFTWARE	WWW.RSWSOFTWARE.COM	508.435.8000	27
SD 2000	WWW.SDEXPO.COM		17
SHIFT4	WWW.SHIFT4.COM	800.265.5795	13
SITEHOSTING.NET	WWW.SITEHOSTING.NET	888.463.6168	37
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	212.460.8406	39

Get Your Own Subscriptions to the Finest Technical Journals in the Industry!

1-800-513-7111

www.sys-con.com

## Able Solutions

Enter the realm of browsable store building and administration – from your browser. Build “your\_site.com” with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing.

11700 NE 95th Street, Suite 100, Vancouver, WA  
[www.ablecommerce.com](http://www.ablecommerce.com) • 360 253-4142

## Adhost Internet Advertising

Adhost provides complete web hosting solutions for over twelve hundred business clients. Small firms to multi-nationals, startups to long established companies - every business with which we do business receives the unparalleled level of service and range of products that has set Adhost Internet apart from the pack since 1995.

400 108th Avenue NE, Suite 700, Bellevue, WA 98004  
[www.adhost.com](http://www.adhost.com) • (888) ADHOST-1

## Catouzer

Catouzer develops web-based intranet and Customer Relationship Management software solutions. With Synergy 2.0, Catouzer continues its lead in providing secure web-based work environments. ColdFusion developers now have the most advanced framework to develop secure web-based projects.

[www.catouzer.com](http://www.catouzer.com) • 604 662-7551

## Data Return Corporation

Data Return offers extensive support for customers utilizing ColdFusion from Allaire. With customers delivering over 50,000 user sessions per day, we know how to provide high availability solutions for this advanced application server. Our technical support staff has extensive experience in coding custom ColdFusion Tags as well as managing Microsoft SQL server. We also support CyberCash for customers interested in real-time credit card processing. Our combination of support and experience offer an ideal environment for deploying your applications developed for ColdFusion.

801 Stadium Dr., Ste 117, Arlington, TX 76011  
[www.datareturn.com](http://www.datareturn.com) • 800 767-1514

## digitalNATION - a VERIO company

digitalNATION, VERIO's Advanced Hosting Division, is the world's leading provider of dedicated server hosting, with over 1,650 servers online today. dN's superior connected network and service abilities have earned dN a solid reputation as a first-choice provider of dedicated server solutions (Sun, Windows NT, Linux and Cobalt). digitalNATION has been providing online and network services for over six years. One of the first ISPs to provide dedicated servers running Microsoft Windows NT, the dN staff has unparalleled experience in this industry.

5515 Cherokee Ave, Alexandria, VA 22312-2309  
[www.dedicatedserver.com](http://www.dedicatedserver.com) • 703 642-2800

## Ektron

Ektron supports the next-generation needs of e-businesses by providing dynamic Web infrastructure solution tools designed for use by nontechnical staff. Ektron's flagship offering, eContentManager, gives staff members across an organization the hands-on ability to make real-time additions and updates to Web content without requiring knowledge of a programming language -- while still allowing for centralized administrative control and security. With competitive advantages such as ease-of-integration and drag & drop everything, Ektron is looking to provide these empowering products to customers, resellers and integrators.

5 Northern Blvd., Suite 6, Amherst, NH 03031  
[www.ektron.com](http://www.ektron.com) • 603-594-0249

## Enhanced Technologies

Enhanced Technologies is a state of the art Web Consulting, Development, and Marketing provider built on the principles of ease of use, strength of service, and power of presentation. ETI is not just another firm offering Web development services. We are a professional Web Development company with many more services and capabilities than most of our competitors. We offer HTML (Hypertext) design, Web graphics design, and CGI programming, but we also offer 3D graphics, inline, Java, and Macro-media Director animation.

6422 Grovedale Dr., Suite 301E, Alexandria, VA 22310  
[www.enhtech.com](http://www.enhtech.com) • 800-368-3249

## EnterAct

EnterAct is the Business Services Group of 21st Century Telecom - Chicago's only single-source, facilities-based provider of bundled telecommunications. Combine this with EnterAct's excellent customer service and technical support and it's easy to see why EnterAct is the premier Internet Service Provider in Illinois.

407 S. Dearborn, 6th Floor, Chicago, IL 60605  
[www.enteract.com](http://www.enteract.com) • 312-955-3000

## Enterprise USA, Inc.

Enterprise USA is a software product and computer consulting firm based in the Metropolitan Detroit area of Michigan, with offices in Farmington and Lansing. Enterprise USA specializes in providing computer consultants for Internet development, Linux and the Microsoft product line. In particular they provide supplemental staffing of computer programmers, help desk professionals, network integration specialists, and project managers.

33425 Grand River Ave, Suite 201, Farmington, MI 48335  
[www.enterpriseUSA.com](http://www.enterpriseUSA.com) • 248 888-1473

## Eprise Corporation

If your customers are looking for a content management solution, Eprise Participant Server FastStart Kit for Allaire ColdFusion developers can save you time and resources. Participant Server is a flexible-content management framework that enhances high-value business relationships through the delivery of timely, targeted, Web-based communications.

1671 Worcester Road, Framingham, MA 01701  
[www.eprise.com](http://www.eprise.com) • 800 274-2814



Highlight your website with  
**infoboard**  
NT and UNIX  
Cold Fusion Hosting  
Development Consulting  
Oracle, Informix, MSSQL, E-Commerce Plug-ins  
1-800-514-2297  
sales@infoboard.com  
www.infoboard.com

## Intermedia, Inc.

Our advanced virtual hosting packages (powered by Microsoft Windows NT and Internet Information Server 4.0) offer an environment supporting everything today's advanced Web developer or sophisticated client could ask for. Complete ODBC support is available on plans B and C. We support Microsoft Index Server on all hosting plans.

953 Industrial Avenue, Suite 121, Palo Alto, CA 94303  
[www.intermedia.net](http://www.intermedia.net) • 650 424-9935

## ISITE Design

ISITE Design is a customer-focused New Media firm celebrating two years of serving the Portland, San Francisco and Los Angeles markets. Our diverse team provides a full range of services for our clients. From conceptualization and original art generation to database integration and online marketing, our team is poised to provide your organization with the highest level of customer service and results.

615 SW Broadway Ste. 200, Portland, OR 97205  
[www.isitedesign.com](http://www.isitedesign.com) [www.buyspectra.com](http://www.buyspectra.com) • 888-269-9103

## NetFast

NetFast strives to become the foremost leader in Web application software. Our applications empower and enable function-building efforts of Web developers and designers. These ready-to-use software components can be bundled into existing Intranet and Internet sites to increase usability, drive traffic and differentiate online offerings. The fusion of application development and hosting provides substantial revenue opportunities for VARs and integrators of NetFast products.

6699 Port West Drive, Suite 130, Houston, TX 77024  
[www.netfast.net](http://www.netfast.net) • 800 362-9004

## On-Line Data Solutions

CoolFusion.com is dedicated to providing unique and powerful add-on solutions for ColdFusion development and implementation. The site is hosted and maintained by On-Line Data Solutions, Inc. - a leader in ColdFusion integration. Our ColdFusion integration products line is called iFusion - a combination of "infuse" and ColdFusion. For information about our flagship product, iFusion Mail Server, we invite you to read the online information (where you can also download the latest beta) and join the iFusion Mail Server support list.

24 Elm Street, Centereach, NY 11720-1706  
[www.coolfusion.com](http://www.coolfusion.com) • 516 737-4668

## RSW Software

RSW Software is a wholly owned subsidiary of Teradyne, Inc., and specializes in Web application testing software. Established with the goal of providing best-in-class testing products, RSW offers a suite of products called the e-TEST Suite, which automates the process of testing business-critical Internet and intranet applications.

44 Spring Street, Second Floor, Watertown, MA 02172  
[www.rswsoftware.com](http://www.rswsoftware.com) • 508 435-8000

## SHIFT4 Corporation

Shift4 Corporation is a leading provider of credit card and transaction processing software utilized by merchants and application developers in various industries. Shift4 products facilitate the point-of-sale and accounting functions associated with electronic payment media, including credit cards, debit cards, purchase cards, specialty cards, private label cards and electronic check clearing. More than 2,000 customers worldwide utilize Shift4 software to process over 85 million credit card transactions annually.

8691 W. Sahara Ave., Las Vegas, NV 89117-5830  
[www.shift4.com](http://www.shift4.com) • 800 265-5795

## Sitehosting.NET

Successful electronic commerce starts at SiteHosting.net; a division of Dynatek Infoworld, Inc., which provides total Web development services. We offer personal and efficient customer service with reliability at value prices. All our plans include access to SSL (Secure Socket Layer). We support ColdFusion, Active Server Pages, Real Audio/Video, Netshow Server, and more. Our hosting price starts at \$14.95/month.

13200 Crossroads Parkway North, Suite 360, City of Industry, CA 91746  
[www.sitehosting.net](http://www.sitehosting.net) • 877 684-6784

## Virtualscape

Why host with Virtualscape? Nobody else on the Internet understands what it takes to host ColdFusion like we do. Virtualscape is the leader in advanced Web site hosting. From Fortune 500 extranets to e-commerce sites and more, developers recognize our speed, stability, reliability and technical support.

215 Park Avenue South, Suite 1905, New York, NY 10003  
[www.virtualscape.com](http://www.virtualscape.com) • 212 460-8406

To place an ad in the  
ColdFusion Marketplace  
contact Robyn Forma at  
914 735-0300



## Allaire Acquires Valto Systems

(Cambridge, MA) – Allaire Corporation has acquired Valto Systems, a pioneer in EJB server technology. The acquisition further expands Allaire's position in the emerging e-business platform market.

Under the terms of the agreement, Valto's employees will immediately be a part of Allaire Corporation. Allaire plans to continue selling Valto's product, Ejpt 1.2, and to release a public beta of the next generation of Ejpt this month.

[www.allaire.com](http://www.allaire.com) [www.valto.com](http://www.valto.com)

## Allaire Partners with MERANT

(Mountain View, CA / Newbury, England) – MERANT has announced an agreement with Allaire Corporation to integrate MERANT's DataDirect technology

with the new Linux versions of ColdFusion, empowering Allaire customers with secure high-availability data access and integration across Linux-based Web applications.

[www.allaire.com](http://www.allaire.com) [www.merant.com](http://www.merant.com)

## Allaire Adds New VP to Executive Management Team

(Cambridge, MA) – Allaire Corporation has announced the addition of George Favaloro to its senior management team. With nearly 20 years of experience, this industry veteran will provide considerable depth and experience to Allaire's executive roster.

As vice president of business development, Favaloro will manage Allaire's emerging business strategy formulation group, the mergers and acquisitions team,

and strategic partnerships account team.

[www.allaire.com](http://www.allaire.com)

## Atomic Software Promotes Interface for ColdFusion

(Alpharetta, GA) – For its Internet payment service, iAuthorizer, Atomic Software announces the

availability of a custom tag for

ColdFusion. The tag allows a Web developer to integrate the real-time credit card processing that the iAuthorizer service provides to e-commerce-enable a Web site.

The interface, called CF\_iAuthorizerCC, is available for download at Allaire's developers exchange site.

[www.iAuthorizer.com](http://www.iAuthorizer.com)

## AbleCommerce Announces TopSites.com

(Vancouver, WA) – AbleCommerce has launched TopSites.com, a showcase of the Web's top stores.

[www.topsites.com](http://www.topsites.com) provides a dynamic search engine or "mall" of the wide range of shopping sites that have been built and are administered with AbleCommerce. The listing is a free service to AbleCommerce storeowners.

A few of the well-known companies listed at TopSites.com are Bushnell, CITGO, Cummins

Engine, Dixie Chicks, Edwin Watts Golf Shops, Garth Brooks, George Strait, New England Patriots, Smith & Wesson, and Victorinox Swiss Army Knives.

[www.ablesolutions.com](http://www.ablesolutions.com)



# Online Data Solutions

[www.coolfusion.com](http://www.coolfusion.com)

# BiznizWeb

[www.webpublishingtools.com](http://www.webpublishingtools.com)

# Infoboard

[www.infoboard.com](http://www.infoboard.com)

# **Catouzer**

**[www.catouzer.com](http://www.catouzer.com)**

# NT WEB HOSTING Intermedia. NET

TAKE  
control  
Web of your  
Site

TEST DRIVE FREE DEMO ACCOUNT



**Unlimited** Email accounts  
**Unlimited** traffic/bandwidth

**New!** Clockwork Mail<sup>™</sup>  
Manage email from your browser

FOR  
**FREE**  
SET UP  
USE PROMO CODE  
**CFDJ**

NT Hosting with IIS 4.0  
MS SQL Server 7.0, Access, FoxPro  
Active Server Pages  
Front Page 98 and 2000  
Cold Fusion 4.01  
SSL Secure Server  
Shopping Carts: StoreFront, Drumbeat  
ECommerce  
CyberCash, PaymentNet  
WebTrends Statistics

Reseller Program Available

## Instant Control!

Add domains  
Add disk space  
Add email accounts  
Add mailing lists  
Register Cold Fusion Custom Tags  
Register ODBC  
Register Active X DLL and OCX

sign up online  
and activate your account in 10 minutes

**www.intermedia.net**

**800.379.7729**



Intermedia.NET, Inc.

953 Industrial Ave  
Palo Alto CA 94303

sales@intermedia.net